EFFICIENT NEURAL MACHINE TRANSLATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND
ELECTRONIC ENGINEERING
OF THE UNIVERSITY OF HONG KONG
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Jiatao Gu
June 2018

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality for the degree of Doctor of Philosophy.

(Victor O.K. Li) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality for the degree of Doctor of Philosophy.

(Kyunghyun Cho)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality for the degree of Doctor of Philosophy.

(Yizhou Yu)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality for the degree of Doctor of Philosophy.

(Y. C. Wu)

Approved for the University of Hong Kong Committee on Graduate Studies

Abstract

The dream of automatic translation that builds the communication bridge between people from different civilizations dates back to thousands of years ago. For the past decades, researchers devoted to proposing practical plans, from rule-based machine translation to statistical machine translation. Till recent years, with the general success of artificial intelligence (AI) and the emergence of neural network models, a.k.a. Deep learning, neural machine translation (NMT), as the new generation of machine translation framework based on sequence-to-sequence learning has achieved the state-of-the-art and even human-level translation performance on a variety of languages.

The impressive achievements brought by NMT mainly owes to its deep neural network structures with massive amounts of parameters, which can be efficiently tuned from vast volume of parallel data in the order of tens or hundreds of millions of sentences. Unfortunately, in spite of the success neural systems also bring about new challenges to machine translation, in which one of the central problems is efficiency. The efficiency issue involves two aspects: (1) NMT is data-hungry because of its vast size of parameters, which makes training a reasonable model difficult in practice for low resource cases. For instance, most of the human languages do not have enough parallel data with other languages to learn an NMT model; Moreover, documents in specialized domains such as law or medical files usually contain tons of professional translations, leading to less efficient for NMT to learn from; (2) NMT is slow for computation compared to conventional methods due to its deep structure and limitations of the decoding algorithms. Especially the low efficiency at inference time profoundly affects the real-life application and the smoothness of the communication. In some cases like video conference, we also hope the neural system to translate at real-time which, however, is difficult for the existing NMT models.

This dissertation attempts to tackle these two challenges, respectively. Contributions are twofold: (1) we address the data-efficiency challenges presented by existing NMT models and introduce insights based on the characteristics of the data, which includes (a) developing the copy-mechanism to target on rote memories in translation and general sequence-to-sequence learning; (b) using a non-parametric search-engine to guide the NMT system to perform well in special domains; (c) inventing a universal NMT system to extremely low resource languages; (d) extending the universal NMT system to be able to efficiently adapt to new languages by combing with meta-learning. (2) for the decoding-efficiency challenges, we develop novel structures and learning algorithms (a) recasting the decoding of NMT in a trainable manner to achieve state-of-the-art performance with less time; (b) inventing the non-autoregressive NMT system which enables translation in entirely parallel; (c) developing the NMT model that learns to translate in real-time using reinforcement learning.

(449 words)

Acknowledgments

Thank you very much!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 A Brief Review of Neural Machine Translation

1.2 Efficient Neural Machine Translation

1.3 Thesis Outline

Chapter 2

Background: Neural Machine Translation

In this chapter, we will introduce some background knowledge about neural machine translation on three aspects, namely, *modeling*, *training*, and *decoding*

2.1 Modeling

We start from discussing the building blocks that forms the most recent Neural Machine Translation (NMT) models. For all different architectures, NMT can always be seen as a conditional language model. The introduction of the attention mechanism becomes the key which enables the NMT to achieve the state-of-the-art performance.

2.1.1 Neural Language Modeling

Before discussing the details of nerual machine translation, we first come to the general topic of generating sentence using neural networks, a.k.a. neural language modeling. The research of language modeling with neural networks dates back to (NNLM, Bengio et al., 2003) where the authors used a feed-forward network to predict next words with a fixed window size of words as input. In general, when we consider the problem of natural language modeling, a simplified view is to model languages (e.g. sentences) in a generative

framework: let V the vocabulary of all possible tokens, for a sentence $Y = \{y_1, y_2, ..., y_T\}$, $y_t \in V$, the language model finds a set of parameters θ to represent the reconstruction probability $p(Y; \theta) = p(y_1, y_2, ..., y_T; \theta)$.

Autoregressive Language Model Directly modeling the probability of the entire sentence Y is hard and intractable. Therefore aside from some research works, most of previous efforts reformulates the probability in an autoregressive way, that is,

$$p(Y; \theta) = \prod_{t=1}^{T+1} p(y_t | y_{0:t-1}; \theta), \tag{2.1}$$

where special tokens y_0 (e.g. $\langle \text{bos} \rangle$) and y_{T+1} (e.g. $\langle \text{eos} \rangle$) are used to represent the beginning and end of all target sentences. The intuiative view is that the language model shows that each generated tokens y_t is conditional to all previous generated tokens.

Parameterization We are interested in parameterizing (approximating) these conditional probabilities using neural networks, which can be feed-forward networks, convolutional networks, or recurrent neural networks (RNNLM, Mikolov et al., 2010). Take the RNNLM – a generic formulation for language model – as an example. For each time-step t, the whole history $y_{0:t-1}$ is summarized as:

$$h_t = f(\epsilon_I[y_{t-1}], h_{t-1}; \theta_f), t \in [1, T] \tag{2.2}$$

where ϵ_I is an embedding matrix and each row maps a fixed-length vector for each input words, f is the recurrent networks used to summarize the history. In practise, instead of using the vanilla RNN, RNNs with gated units are often used to capture long-term dependency in language modeling, for instance, Long-short term memory (LSTM, Hochreiter and Schmidhuber, 1997) or Gatetd Recurrent Units (GRU, Cho et al., 2014b). Therefore, we can compute the output probability of y_t by

$$p(y_t | y_{0:t-1}; \theta) = \underset{y_t \in V}{\text{softmax}} \left(\epsilon_O[y_t] \cdot h_t^T \right) \tag{2.3}$$

where $\text{softmax}(x) = e^x / \sum_{x'} e^{x'}$ is to ensure the probabilities sum to 1, and ϵ_O is the output weights which are sometimes tied with ϵ_I. More complicated structure with multiple layers of networks can be used to enhance the modeling ability.

2.1.2 Sequence-to-Sequence Learning

The neural language modeling tells how likely to a natural language sentence can be generated from nowhere, which however, is not practical in real applications. In most cases, what we are interested in is to generate meaningful sentences with certain conditions, which includes not only machine translation – the main focus of this thsis – but many other tasks like summarization, captioning, question answering, and response generation in dialog system. Fortunately, it is easy to borrow the experience of language modeling by assuming a conditional language modeling. It is also known as sequence-to-sequence (SEQ2SEQ) learning when the conditioning inputs are also sequences, e.g. neural machine translation.

Neural Machine Translation as SEQ2SEQ Learning Let us denote $X = \{x_1, \ldots, x_{T'}\}$ and $Y = \{y_1, \ldots, y_T\}$ to denote source and target sentences in machine translation, respectively. Similar to language modeling, neural machine translation models the target sentence given the source sentence as a conditional autoregressive language model:

$$p(Y|X) = \prod_{t=1}^{T+1} p(y_t|y_{0:t-1}, x_{1:T'}; \theta), \tag{2.4}$$

We can use the same output function as Eq. 2.3 for each probability where in SEQ2SEQ learning, and the hidden states h_t summarizes both the decoded and the source words

$$h_t = f^{\text{DEC}}(\epsilon_I[y_{t-1}], h_{t-1}, c_t(x_{1:T'}); \theta_f), t \in [1, T+1] \tag{2.5}$$

where $c_t(x_{1:T'})$ is the context representation of the source sentence X. For instance, a simple implementation proposed in Cho et al. (2014b); Sutskever et al. (2014) is to use another RNN to encode the source words into a series of vector representations $s_1, ..., s_{T'}$ as

$$s_\tau = f^{\text{ENC}}\left(\epsilon_E[x_\tau], s_{\tau-1}; \theta_e\right), \tau \in [1, T'] \tag{2.6}$$

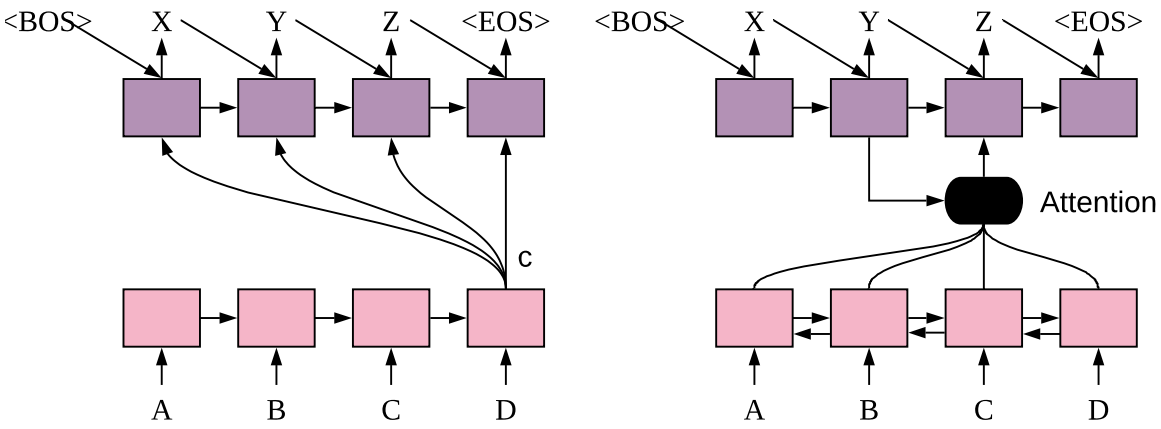


Figure 2.1: An illustration of the comparison between the conventional SEQ2SEQ learning and SEQ2SEQ with attention mechanism for translating "A B C D $\rightarrow$ X Y Z ".

where each source word x_τ is projected into a continuous embedding space with ϵ_E. Then, it is possible to use the last state $s_{T'}$ as the context c considering it has compressed all the information we need to get the correct translation.

Borrowing the concepts from information theory, we usually call the f^{ENC} and f^{DEC} in SEQ2SEQ learning as the encoder and decoder, respectively. Intuitively in NMT, we first encode the whole source sentence X into a fixed length vector $c = s_{T'}$, and then we decode the translation from it.

2.1.3 Attention Mechanism

Obviously, it is difficult to encode all the necessary information for translation into a fixed size vector, especially when the source sentence gets long. To release such burden, the attention mechanism was then introduced in Bahdanau et al. (2014), which was also inspired from the concept of "alignment" between source and target words in statistical machine translation. Instead of using a single vector for the whole decoding path, the attention mechanism uses a dynamically changing context c_t in the decoding process. A natural option is to represent c_t as the weighted sum of the source hidden states, i.e.

$$c_t = \sum_{\tau=1}^{T'} \alpha_{t\tau} s_\tau, \quad \alpha_{t\tau} = \underset{\tau}{\text{softmax}} \left(f^{\text{ATT}}(h_{t-1}, s_\tau; \theta_a) \right), \tag{2.7}$$

where α are the attention weights which is treated as soft-alignment, and f^{ATT} is the function that shows the correspondence strength for attention, which can be either approximated with a multi-layer neural network or the inner-products between h_{t-1} and s_τ. Note that in Bahdanau et al. (2014), the source sentence is encoded with a bidirectional instead of an unidirectional RNN discussed above in Eq. 2.6, making each hidden state s_τ aware of the contextual information from both ends.

The attention mechanism nicely solved the information compression problem in machine translation, and is the key for NMT to be able to achieve the state-of-the-art performance. See a comparison with a conventional SEQ2SEQ model in Fig. 2.1. For the following sections and chapters, without special notification, we in default consider the RNN-based SEQ2SEQ model with attention mechanism as the basic NMT model we used for exploration.

2.2 Training

2.2.1 Parallel Corpora

Training requires large amount of parallel corpus

We will address this problem in the later chapters

2.2.2 Maximum Likelihood Learning

We train an NMT model, or equivalently estimate $\theta = \{\theta_f, \theta_e, \theta_a, \epsilon_E, \epsilon_I, \epsilon_O\}$, by maximizing the log-probability of the target translation Y given a source sentence X. That is, we maximize the log-likelihood function:

$$\mathcal{L}^{\text{ML}}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n+1}\log p(y_t^n|y_{0:t-1}^n, x_{1:T'}^n;\theta), \tag{2.8}$$

given a training set consisting of N parallel sentence pairs. One of the fascinate things about NMT and deep learning is that, once we got the formulation of a differentiable objectives, the whole model parameters – no matter how complex you choose your model to be – can be

trained end-to-end using stochastic gradient decent (SGD) via back-propagtation (Rumelhart et al., 1986). In practise, we can use modified grandients outputted from an optimizer function, such as Adadelta (Zeiler, 2012) or Adam (Kingma and Ba, 2014) to achieve a faster convergence speed and better performance .

It is important to note that this maximum likelihood learning does not take into account how a trained model would be used. Rather, it is only concerned with learning a distribution over all possible translations.

2.2.3 Advanced Learning Algorithms

Unfortunately, MLE training has some limitations. (Wiseman and Rush, 2016; Shen et al., 2015; Bahdanau et al., 2016; Ranzato et al., 2015)

2.2.4 Multilingual Training of Neural Machine Translation

2.3 Decoding

Once the model is trained, either by maximum likelihood learning or by any other recently proposed algorithms above, we can let the model translate a given sentence by finding a translation that maximizes

$$\hat{Y} = \underset{Y}{\operatorname{argmax}} \log p(Y|X;\theta) = \underset{y_1,\dots,y_T}{\operatorname{argmax}} \sum_{t=1}^{T+1} \log p(y_t|y_{0:t-1}, x_{1:T'};\theta), \tag{2.9}$$

where $\theta = \{\theta_f, \theta_e, \theta_a, \epsilon_E, \epsilon_I, \epsilon_O\}$, $y_0 = \langle \mathrm{bos} \rangle$ and $y_{T+1} = \langle \mathrm{eos} \rangle$. This process is so-called *decoding*, which is, however, computationally intractable as the number of possible solutions grows exponentially when the decoding length T increases. Although the training algorithms ensure a good approximation for the translation distribution, it does not tell how to find the translation given the distribution function. It is a usual practice to resort to approximate decoding algorithms.

2.3.1 Greedy Decoding

One such approximate decoding algorithm is greedy decoding. In greedy decoding, we follow the conditional dependency path and pick the symbol with the highest conditional probability so far at each step. This is equivalent to picking the best symbol one at a time from left to right in conditional language modeling. A decoded translation of greedy decoding is $\hat{Y} = (\hat{y}_1, \ldots, \hat{y}_T)$, where

$$\hat{y}_t = \underset{y \in V}{\text{argmax}} \log p(y | \hat{y}_{0:t-1}, x_{1:T'}; \theta). \tag{2.10}$$

Despite its preferable computational complexity $O(|V| \times T)$, greedy decoding has been over time found to be undesirably sub-optimal.

2.3.2 Beam Search

A better option to improve the translation quality is to perform beam search. Unlike greedy decoding which keeps only a single hypothesis, beam search keeps $K(> 1)$ hypotheses simultaneously during decoding. That is, at each time step t, beam search picks K hypotheses with the highest scores from the cached K prefix sequences (beam)[1]:

$$\mathcal{C}_{t-1} = \{\tilde{y}^{(1)}_{0:t-1}, \tilde{y}^{(2)}_{0:t-1}, ..., \tilde{y}^{(K)}_{0:t-1}\} \tag{2.11}$$

by evaluating the cumulative conditional probabilities of all candidate hypotheses to obtain the next round of sequences:

$$\mathcal{C}_t = \{\tilde{y}^{(1)}_{0:t}, \tilde{y}^{(2)}_{0:t}, ..., \tilde{y}^{(K)}_{0:t}\} = \underset{y_t \in V, y_{0:t-1} \in \mathcal{C}_{t-1}}{\text{argsort}^K} \sum_{t'=0}^{t} \log p(y_{t'} | y_{0:t'-1}, x_{1:T'}; \theta). \tag{2.12}$$

This process repeats until one or some of the hypotheses hits the end-of-sentence symbol ($\langle \text{eos} \rangle$). Each ended sequence $\tilde{Y}$ will be selected from the cache and in the meantime, the beam size decreases $K = K - 1$. When all hypotheses terminate, it returns the best hypothesis from the chosen sequences $\tilde{Y}_1, ..., \tilde{Y}_K$ by a re-ranking function. For instance, a

[1] We assume for the initial step $t = 0$, we only has one symbol $\mathcal{C}_0 = \{ \langle \text{bos} \rangle \}$

simple but effective way is to use the log-probability factored by the decoding length:

$$\hat{Y} = \underset{\tilde{Y}_1,\ldots,\tilde{Y}_K}{\operatorname{argmax}} \left(\frac{1}{|Y|}\right)^{\alpha} \cdot \log p(Y|X;\theta), \tag{2.13}$$

where we use $|Y|$ to show the number of tokens in the decoded sentence, and $\alpha \in [0, 1]$. When we selected $\alpha = 1$, the re-ranking score is equivalent to a normalized log-likelihood, or so-called perplexity. In practise, we can choose $\alpha \approx 0.6$ which can greatly help to avoid generating too short sentences.

Despite its superior performance compared to greedy decoding, the computational complexity grows linearly w.r.t. the size of beam K, which makes it less preferable especially in the production environment.

2.3.3 Noisy Parallel Decoding

Another alternative we can consider from is the noisy, parallel decoding (NPD) algorithm proposed in Cho (2016). The main idea behind NPD algorithm is that a better translation with a higher log-probability may be found by injecting unstructured noise in the transition function of the decoder. That is,

$$h_t = f^{\text{DEC}}(\epsilon_I[y_{t-1}], h_{t-1} + n_t, c_t(x_{1:T'}); \theta_f), t \in [1, T + 1] \tag{2.14}$$

where $n_t \sim \mathcal{N}(0, \sigma_0^2/t^2)$ is the time-dependent noise. In practise, we can also other ways to inject randomness in the inner representation of the decoding process, and run greedy decoding based on the "noisy" inputs. NPD avoids potential degradation of translation quality by running such a noisy greedy decoding process multiple times in parallel. An important lesson of NPD algorithm is that there exists a decoding strategy with the asymptotically same computational complexity that results in a better translation quality, and that such a better translation can be found by manipulating the hidden state of the recurrent network.

2.4 Neural Machine Translation without RNNs

Since the entire target translation is known at training time, the calculation of later conditional probabilities (and their corresponding losses) does not depend on the output words chosen during earlier decoding steps. Even though decoding must remain entirely sequential during inference, models can take advantage of this parallelism during training. One such approach replaces recurrent layers in the decoder with masked convolution layers Kalchbrenner et al. (2016); Gehring et al. (2017) that provide the causal structure required by the autoregressive factorization.

2.4.1 The Transformer Model

A recently introduced option which reduces sequential computation still further is to construct the decoder layers out of self-attention computations that have been causally masked in an analogous way. The state-of-the-art Transformer network takes this approach, which allows information to flow in the decoder across arbitrarily long distances in a constant number of operations, asymptotically fewer than required by convolutional architectures (Vaswani et al., 2017).

Part I

Data-Efficient Neural Machine Translation

Chapter 3

Incorporating Copying Mechanism in Sequence-to-Sequence Learning

As introduced in Chapter 2, sequence-to-sequence (SEQ2SEQ) learning has achieved remarkable success in various natural language processing (NLP) tasks, including but not limited to Machine Translation (Cho et al., 2014b; Bahdanau et al., 2014), Syntactic Parsing (Vinyals et al., 2015b), Text Summarization (Rush et al., 2015) and Dialogue Systems (Vinyals and Le, 2015). Adding the attention mechanism (Bahdanau et al., 2014) has led to significant improvement on the performance of various tasks (Shang et al., 2015; Rush et al., 2015). Different from the canonical encoder-decoder architecture, the attention-based SEQ2SEQ model revisits the input sequence in its raw form (array of word representations) and dynamically fetches the relevant piece of information based mostly on the feedback from generation of the output sequence.

In this chapter, we explore another mechanism important to the human language communication, called the "copying mechanism". Basically, it refers to the mechanism that locates a certain segment of the input sentence and puts the segment into the output sequence. For example, in the following two dialogue turns we observe different patterns in which some subsequences (colored blue) in the response (R) are copied from the input utterance (I):

I: Hello Jack, my name is Chandralekha.
R: Nice to meet you, Chandralekha.

I: This new guy doesn't perform exactly as we expected.
R: What do you mean by "doesn't perform exactly as we expected"?

Both the canonical encoder-decoder and its variants with attention mechanism rely heavily on the representation of "meaning", which might not be sufficiently inaccurate in cases in which the system needs to refer to sub-sequences of input like entity names or dates. In contrast, the copying mechanism is closer to the rote memorization in language processing of human being, deserving a different modeling strategy in neural network-based models. We argue that it will benefit many Seq2Seq tasks to have an elegant unified model that can accommodate both understanding and rote memorization. Towards this goal, we propose COPYNET, which is not only capable of the regular generation of words but also the operation of copying appropriate segments of the input sequence. Despite the seemingly "hard" operation of copying, COPYNET can be trained in an end-to-end fashion. Our empirical study on both synthetic datasets and real world datasets demonstrates the efficacy of COPYNET.

3.1 COPYNET

From a cognitive perspective, the copying mechanism is related to rote memorization, requiring less understanding but ensuring high literal fidelity. From a modeling perspective, the copying operations are more rigid and symbolic, making it more difficult than soft attention mechanism to integrate into a fully differentiable neural model. In this section, we present COPYNET, a differentiable Seq2Seq model with "copying mechanism", which can be trained in an end-to-end fashion with just gradient descent.

3.1.1 Model Overview

As illustrated in Figure 3.1, COPYNET is still an encoder-decoder (in a slightly generalized sense). The source sequence is transformed by **Encoder** into representation, which is then

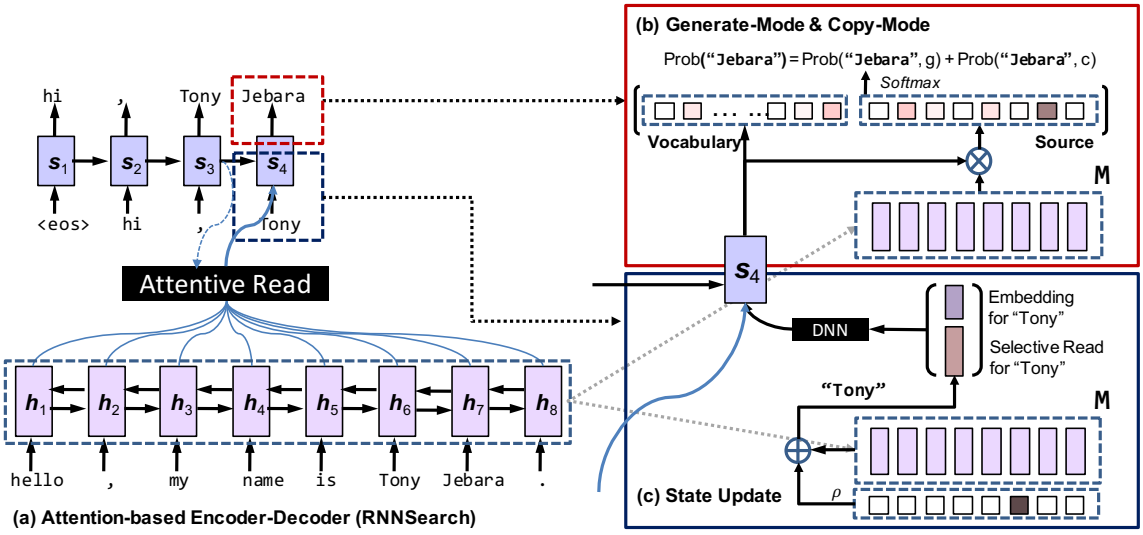


Figure 3.1: The overall diagram of COPYNET. For simplicity, we omit some links for prediction (see Sections 3.2 for more details).

read by **Decoder** to generate the target sequence.

Encoder: Same as in (Bahdanau et al., 2014), a bi-directional RNN is used to transform the source sequence into a series of hidden states with equal length, with each hidden state $\mathbf{h}_t$ corresponding to word x_t. This new representation of the source, $\{\mathbf{h}_1, ..., \mathbf{h}_{T_S}\}$, is considered to be a short-term memory (referred to as $\mathbf{M}$ in the remainder of the paper), which will later be accessed in multiple ways in generating the target sequence (decoding).

Decoder: An RNN that reads $\mathbf{M}$ and predicts the target sequence. It is similar with the canonical RNN-decoder in (Bahdanau et al., 2014), with however the following important differences

- **Prediction:** COPYNET predicts words based on a mixed probabilistic model of two modes, namely the **generate-mode** and the **copy-mode**, where the latter picks words from the source sequence (see Section 3.1.2);
- **State Update:** the predicted word at time $t-1$ is used in updating the state at t, but COPYNET uses not only its word-embedding but also its corresponding location-specific hidden state in $\mathbf{M}$ (if any) (see Section 3.1.3 for more details);

- **Reading M:** in addition to the attentive read to M, COPYNET also has"selective read" to M, which leads to a powerful hybrid of content-based addressing and location-based addressing (see both Sections 3.1.3 and 3.1.4 for more discussion).

3.1.2 Prediction with Copying and Generation

We assume a vocabulary $\mathcal{V} = \{v_1, ..., v_N\}$, and use UNK for any out-of-vocabulary (OOV) word. In addition, we have another set of words $\mathcal{X}$, for all the *unique* words in source sequence $X = \{x_1, ..., x_{T_S}\}$. Since $\mathcal{X}$ may contain words not in $\mathcal{V}$, copying sub-sequence in X enables COPYNET to output some OOV words. In a nutshell, the instance-specific vocabulary for source X is $\mathcal{V} \cup \text{UNK} \cup \mathcal{X}$.

Given the decoder RNN state $\mathbf{s}_t$ at time t together with $\mathbf{M}$, the probability of generating any target word y_t, is given by the "mixture" of probabilities as follows

$$p(y_t|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M}) = p(y_t, \mathsf{g}|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M}) + p(y_t, \mathsf{c}|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M}) \quad (3.1)$$

where g stands for the generate-mode, and c the copy mode. The probability of the two modes are given respectively by

$$p(y_t, \mathsf{g}|\cdot) = \begin{cases} \frac{1}{Z} e^{\psi_g(y_t)}, & y_t \in \mathcal{V} \\ 0, & y_t \in \mathcal{X} \cap \bar{V} \\ \frac{1}{Z} e^{\psi_g(\text{UNK})} & y_t \notin \mathcal{V} \cup \mathcal{X} \end{cases} \quad (3.2)$$

$$p(y_t, \mathsf{c}|\cdot) = \begin{cases} \frac{1}{Z} \sum_{j:x_j=y_t} e^{\psi_c(x_j)}, & y_t \in \mathcal{X} \\ 0 & \text{otherwise} \end{cases} \quad (3.3)$$

where $\psi_g(\cdot)$ and $\psi_c(\cdot)$ are score functions for generate-mode and copy-mode, respectively, and Z is the normalization term shared by the two modes, $Z = \sum_{v \in \mathcal{V} \cup \{\text{UNK}\}} e^{\psi_g(v)} + \sum_{x \in X} e^{\psi_c(x)}$. Due to the shared normalization term, the two modes are basically competing through a softmax function (see Figure 3.1 for an illustration with example), rendering Eq.(3.1) deviated from the canonical definition of mixture model (McLachlan and Basford,

1988). This is also pictorially illustrated in Figure 3.2. The score of each mode is calculated:

Generate-Mode: The same scoring function as in the generic RNN encoder-decoder (Bahdanau et al., 2014) is used, i.e.

$$\psi_g(y_t = v_i) = \mathbf{v}_i^\top \mathbf{W}_o \mathbf{s}_t, \quad v_i \in \mathcal{V} \cup \text{UNK} \tag{3.4}$$

where $\mathbf{W}_o \in \mathbb{R}^{(N+1)\times d_s}$ and $\mathbf{v}_i$ is the one-hot indicator vector for v_i.

Copy-Mode: The score for "copying" the word x_j is calculated as

$$\psi_c(y_t = x_j) = \sigma\left(\mathbf{h}_j^\top \mathbf{W}_c\right) \mathbf{s}_t, \quad x_j \in \mathcal{X} \tag{3.5}$$

where $\mathbf{W}_c \in \mathbb{R}^{d_h \times d_s}$, and σ is a non-linear activation function, considering that the non-linear transformation in Eq.(3.5) can help project s_t and h_j in the same semantic space. Empirically, we also found that using the $\tanh$ non-linearity worked better than linear transformation, and we used that for the following experiments. When calculating the copy-mode score, we use the hidden states $\{\mathbf{h}_1, ..., \mathbf{h}_{T_S}\}$ to "represent" each of the word in the source sequence $\{x_1, ..., x_{T_S}\}$ since the bi-directional RNN encodes not only the content, but also the location information into the hidden states in $\mathbf{M}$. The location informaton is important for copying (see Section 3.1.4 for related discussion). Note that we sum the probabilities of all x_j equal to y_t in Eq. (3.3) considering that there may be multiple source symbols for decoding y_t. Naturally we let $p(y_t, \mathsf{c}|\cdot) = 0$ if y_t does not appear in the source sequence, and set $p(y_t, \mathsf{g}|\cdot) = 0$ when y_t only appears in the source.

3.1.3 State Update

COPYNET updates each decoding state $\mathbf{s}_t$ with the previous state $\mathbf{s}_{t-1}$, the previous symbol y_{t-1} and the context vector $\mathbf{c}_t$ following Eq. (**??**) for the generic attention-based Seq2Seq model. However, there is some minor changes in the $y_{t-1} \longrightarrow \mathbf{s}_t$ path for the copying mechanism. More specifically, y_{t-1} will be represented as $[\mathbf{e}(y_{t-1}); \zeta(y_{t-1})]^\top$, where $\mathbf{e}(y_{t-1})$ is the word embedding associated with y_{t-1}, while $\zeta(y_{t-1})$ is the weighted sum of hidden states in $\mathbf{M}$ corresponding to y_t

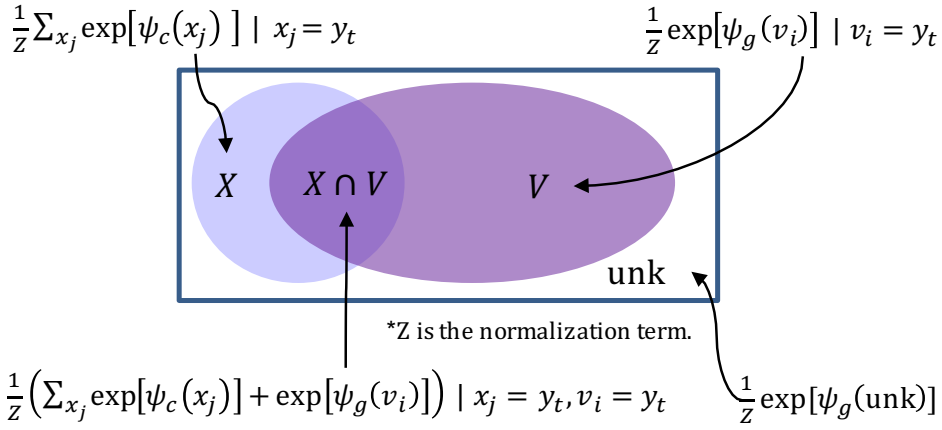


Figure 3.2: The illustration of the decoding probability $p(y_t|\cdot)$ as a 4-class classifier.

$$\zeta(y_{t-1}) = \sum_{\tau=1}^{T_S} \rho_{t\tau}\mathbf{h}_\tau$$
$$\rho_{t\tau} = \begin{cases} \frac{1}{K}p(x_\tau, \mathbf{c}|\mathbf{s}_{t-1}, \mathbf{M}), & x_\tau = y_{t-1} \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

where K is the normalization term which equals $\sum_{\tau':x_{\tau'}=y_{t-1}} p(x_{\tau'}, c|\mathbf{s}_{t-1}, \mathbf{M})$, considering there may exist multiple positions with y_{t-1} in the source sequence. In practice, $\rho_{t\tau}$ is often concentrated on one location among multiple appearances, indicating the prediction is closely bounded to the location of words.

In a sense $\zeta(y_{t-1})$ performs a type of read to $\mathbf{M}$ similar to the attentive read (resulting $\mathbf{c}_t$) with however higher precision. In the remainder of this paper, $\zeta(y_{t-1})$ will be referred to as *selective read*. $\zeta(y_{t-1})$ is specifically designed for the copy mode: with its pinpointing precision to the corresponding y_{t-1}, it naturally bears the location of y_{t-1} in the source sequence encoded in the hidden state. As will be discussed more in Section 3.1.4, this particular design potentially helps copy-mode in covering a consecutive sub-sequence of words. If y_{t-1} is not in the source, we let $\zeta(y_{t-1}) = \mathbf{0}$.

3.1.4 Hybrid Addressing of $\mathbf{M}$

We hypothesize that COPYNET uses a hybrid strategy for fetching the content in $\mathbf{M}$, which combines both content-based and location-based addressing. Both addressing strategies are coordinated by the decoder RNN in managing the attentive read and selective read, as well as determining when to enter/quit the copy-mode.

Both the semantics of a word and its location in X will be encoded into the hidden states in $\mathbf{M}$ by a properly trained encoder RNN. Judging from our experiments, the attentive read of COPYNET is driven more by the semantics and language model, therefore capable of traveling more freely on $\mathbf{M}$, even across a long distance. On the other hand, once COPYNET enters the copy-mode, the selective read of $\mathbf{M}$ is often guided by the location information. As the result, the selective read often takes rigid move and tends to cover consecutive words, including UNKs. Unlike the explicit design for hybrid addressing in Neural Turing Machine (Graves et al., 2014; Kurach et al., 2015), COPYNET is more subtle: it provides the architecture that can facilitate some particular location-based addressing and lets the model figure out the details from the training data for specific tasks.

Location-based Addressing: With the location information in $\{\mathbf{h}_i\}$, the information flow

$$\zeta(y_{t-1}) \xrightarrow{\text{update}} \mathbf{s}_t \xrightarrow{\text{predict}} y_t \xrightarrow{\text{sel. read}} \zeta(y_t)$$

provides a simple way of "moving one step to the right" on X. More specifically, assuming the selective read $\zeta(y_{t-1})$ concentrates on the ℓ^{th} word in X, the state-update operation $\zeta(y_{t-1}) \xrightarrow{\text{update}} \mathbf{s}_t$ acts as "`location` $\leftarrow$ `location+1`", making $\mathbf{s}_t$ favor the $(\ell{+}1)^{th}$ word in X in the prediction $\mathbf{s}_t \xrightarrow{\text{predict}} y_t$ in copy-mode. This again leads to the selective read $\hat{h}_t \xrightarrow{\text{sel. read}} \zeta(y_t)$ for the state update of the next round.

Handling Out-of-Vocabulary Words Although it is hard to verify the exact addressing strategy as above directly, there is strong evidence from our empirical study. Most saliently, a properly trained COPYNET can copy a fairly long segment full of OOV words, despite the lack of semantic information in its $\mathbf{M}$ representation. This provides a natural way to extend the effective vocabulary to include all the words in the source. Although this change is small, it seems quite significant empirically in alleviating the OOV problem. Indeed, for

many NLP applications (e.g., text summarization or spoken dialogue system), much of the OOV words on the target side, for example the proper nouns, are essentially the replicates of those on the source side.

3.2 Learning

Although the copying mechanism uses the "hard" operation to copy from the source and choose to paste them or generate symbols from the vocabulary, COPYNET is fully differentiable and can be optimized in an end-to-end fashion using back-propagation. Given the batches of the source and target sequence $\{X\}_N$ and $\{Y\}_N$, the objectives are to minimize the negative log-likelihood:

$$\mathcal{L} = -\frac{1}{N}\sum_{k=1}^{N}\sum_{t=1}^{T}\log\left[p(y_t^{(k)}|y_{<t}^{(k)}, X^{(k)})\right], \tag{3.7}$$

where we use superscript to index the instances. Since the probabilistic model for observing any target word is a mixture of generate-mode and copy-mode, there is no need for any additional labels for modes. The network can learn to coordinate the two modes from data. More specifically, if one particular word $y_t^{(k)}$ can be found in the source sequence, the copy-mode will contribute to the mixture model, and the gradient will more or less encourage the copy-mode; otherwise, the copy-mode is discouraged due to the competition from the shared normalization term Z. In practice, in most cases one mode dominates.

3.3 Experiments

We report our empirical study of COPYNET on the following three tasks with different characteristics

1. A synthetic dataset on with simple patterns;
2. A real-world task on text summarization;
3. A dataset for simple single-turn dialogues.

3.3.1 Synthetic Dataset

Dataset: We first randomly generate transformation rules with 5∼20 symbols and variables **x** & **y**, e.g.

a b **x** c d **y** e f ⟶ g h **x** m,

with {a b c d e f g h m} being regular symbols from a vocabulary of size 1,000. As shown in the table below, each rule can further produce a number of instances by replacing the variables with randomly generated subsequences (1∼15 symbols) from the same vocabulary. We create five types of rules, including "**x** → ∅". The task is to learn to do the Seq2Seq transformation from the training instances. This dataset is designed to study the behavior of COPYNET on handling simple and rigid patterns. Since the string to repeat are random, they can also be viewed as some extreme cases of rote memorization.

Rule-type	Examples (e.g. **x** = i h k, **y** = j c)
x → ∅	a b c d **x** e f → c d g
x → **x**	a b c d **x** e f → c d **x** g
x → **x x**	a b c d **x** e f → **x** d **x** g
x y → **x**	a b **y** d **x** e f → **x** d i g
x y → **x y**	a b **y** d **x** e f → **x** d **y** g

Experimental Setting: We select 200 artificial rules from the dataset, and for each rule 200 instances are generated, which will be split into training (50%) and testing (50%). We compare the accuracy of COPYNET and the RNN Encoder-Decoder with (i.e. RNNsearch) or without attention (denoted as Enc-Dec). For a fair comparison, we use bi-directional GRU for encoder and another GRU for decoder for all Seq2Seq models, with hidden layer size = 300 and word embedding dimension = 150. We use bin size = 10 in beam search for testing. The prediction is considered correct only when the generated sequence is exactly the same as the given one.

It is clear from Table 3.1 that COPYNET significantly outperforms the other two on all rule-types except "**x** → ∅", indicating that COPYNET can effectively learn the patterns with variables and accurately replicate rather long subsequence of symbols at the proper places.This is hard to Enc-Dec due to the difficulty of representing a long sequence with very high fidelity. This difficulty can be alleviated with the attention mechanism. However

Rule-type	x → ∅	x → x	x → xx	xy → x	xy → xy
Enc-Dec	**100**	3.3	1.5	2.9	0.0
RNNSearch	99.0	69.4	22.3	40.7	2.6
COPYNET	97.3	**93.7**	**98.3**	**68.2**	**77.5**

Table 3.1: The test accuracy (%) on synthetic data.

attention alone seems inadequate for handling the case where strict replication is needed.

A closer look (see Figure 3.3 for example) reveals that the decoder is dominated by copy-mode when moving into the subsequence to replicate, and switch to generate-mode after leaving this area, showing COPYNET can achieve a rather precise coordination of the two modes.

Pattern:

705 502 X 504 339 270 584 556

→

510 771 581 557 022 230 X 115

102 172 862 X 950

* Symbols are represented by their indices from 000 to 999

** Dark color represents large value.

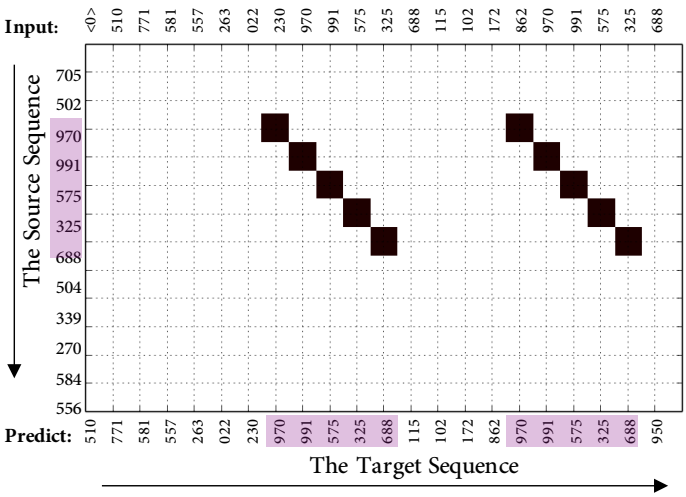


Figure 3.3: Example output of COPYNET on the synthetic dataset. The heatmap represents the activations of the copy-mode over the input sequence (left) during the decoding process (bottom).

3.3.2 Text Summarization

Automatic text summarization aims to find a condensed representation which can capture the core meaning of the original document. It has been recently formulated as a Seq2Seq learning problem in (Rush et al., 2015; Hu et al., 2015), which essentially gives *abstractive* summarization since the summary is generated based on a representation of the document.

In contrast, *extractive* summarization extracts sentences or phrases from the original text to fuse them into the summaries, therefore making better use of the overall structure of the original document. In a sense, COPYNET for summarization lies somewhere between two categories, since part of output summary is actually extracted from the document (via the copying mechanism), which are fused together possibly with the words from the generate-mode.

Dataset: We evaluate our model on the recently published LCSTS dataset (Hu et al., 2015), a large scale dataset for short text summarization. The dataset is collected from the news medias on Sina Weibo[1] including pairs of (short news, summary) in Chinese. Shown in Table 3.2, PART II and III are manually rated for their quality from 1 to 5. Following the setting of (Hu et al., 2015) we use Part I as the training set and and the subset of Part III scored from 3 to 5 as testing set.

Dataset	PART I	PART II	PART III
no. of pairs	2,400,591	10,666	1106
no. of score ≥ 3	-	8685	725

Table 3.2: Some statistics of the LCSTS dataset.

Experimental Setting: We try COPYNET that is based on character (+C) and word (+W). For the word-based variant the word-segmentation is obtained with jieba[2]. We set the vocabulary size to 3,000 (+C) and 10,000 (+W) respectively, which are much smaller than those for models in (Hu et al., 2015). For both variants we set the embedding dimension to 350 and the size of hidden layers to 500. Following (Hu et al., 2015), we evaluate the test performance with the commonly used ROUGE-1, ROUGE-2 and ROUGE-L (Lin, 2004), and compare it against the two models in (Hu et al., 2015), which are essentially canonical Encoder-Decoder and its variant with attention.

It is clear from Table 3.3 that COPYNET beats the competitor models with big margin. Hu et al. (2015) reports that the performance of a word-based model is inferior to a character-based one. One possible explanation is that a word-based model, even with a much larger vocabulary (50,000 words in Hu et al. (2015)), still has a large proportion of OOVs due to the large number of entity names in the summary data and the mistakes in

[1] www.sina.com

[2] https://pypi.python.org/pypi/jieba

Models		ROUGE scores on LCSTS (%)		
		R-1	R-2	R-L
RNN	+C	21.5	8.9	18.6
(Hu et al., 2015)	+W	17.7	8.5	15.8
RNN context	+C	29.9	17.4	27.2
(Hu et al., 2015)	+W	26.8	16.1	24.1
COPYNET	+C	**34.4**	**21.6**	**31.3**
	+W	**35.0**	**22.3**	**32.0**

Table 3.3: Testing performance of LCSTS, where "RNN" is canonical Enc-Dec, and "RNN context" its attentive variant.

word segmentation. COPYNET, with its ability to handle the OOV words with the copying mechanism, performs however slightly better with the word-based variant.

Case Study

As shown in Figure 3.4, we make the following interesting observations about the summary from COPYNET: 1) most words are from copy-mode, but the summary is usually still fluent; 2) COPYNET tends to cover consecutive words in the original document, but it often puts together segments far away from each other, indicating a sophisticated coordination of content-based addressing and location-based addressing; 3) COPYNET handles OOV words really well: it can generate acceptable summary for document with many OOVs, and even the summary itself often contains many OOV words. In contrast, the canonical RNN-based approaches often fail in such cases.

It is quite intriguing that COPYNET can often find important parts of the document, a behavior with the characteristics of extractive summarization, while it often generate words to "connect" those words, showing its aspect of abstractive summarization.

3.3.3 Single-turn Dialogue

In this experiment we follow the work on neural dialogue model proposed in (Shang et al., 2015; Vinyals and Le, 2015; Sordoni et al., 2015), and test COPYNET on single-turn dialogue. Basically, the neural model learns to generate a response to user's input, from the given (input, response) pairs as training instances.

Figure 3.4: Examples of COPYNET on LCSTS compared with RNN context. Word segmentation is applied on the input, where underlined are OOV words. The highlighted words (with different colors) are those words with copy-mode probability higher than the generate-mode. We also provide literal English translation for the document, the golden, and COPYNET, while omitting that for RNN context since the language is broken.

Dataset: We build a simple dialogue dataset based on the following three instructions:

Figure 3.5: Examples on the testing set of DS-II shown as the input text and golden, with the outputs of RNNSearch and CopyNet. Words in red rectangles are unseen in the training set. The highlighted words (with different colors) are those words with copy-mode probability higher than the generate-mode. Green cirles (meaning correct) and red cross (meaning incorrect) are given based on human judgment on whether the response is appropriate.

1. Dialogue instances are collected from Baidu Tieba[3] with some coverage of conversations of real life, e.g., greeting and sports, etc.
2. Patterns with slots like

   $$\texttt{hi, my name is } \mathbf{x} \rightarrow \texttt{hi, } \mathbf{x}$$

   are mined from the set, with possibly multiple responding patterns to one input.
3. Similar with the synthetic dataset, we enlarge the dataset by filling the slots with suitable subsequence (e.g. name entities, dates, etc.)

To make the dataset close to the real conversations, we also maintain a certain proportion of instances with the response that 1) do not contain entities or 2) contain entities not in the input.

Experimental Setting: We create two datasets: DS-I and DS-II with slot filling on 173

[3] http://tieba.baidu.com

collected patterns. The main difference between the two datasets is that the filled substrings for training and testing in DS-II have no overlaps, while in DS-I they are sampled from the same pool. For each dataset we use 6,500 instances for training and 1,500 for testing. We compare COPYNET with canonical RNNSearch, both character-based, with the same model configuration in Section 3.3.1.

	DS-I (%)		DS-II (%)	
Models	Top1	Top10	Top1	Top10
RNNSearch	44.1	57.7	13.5	15.9
COPYNET	**61.2**	**71.0**	**50.5**	**64.8**

Table 3.4: The decoding accuracy on the two testing sets. Decoding is admitted success only when the answer is found exactly in the Top-K outputs.

We compare COPYNET and RNNSearch on DS-I and DS-II in terms of top-1 and top-10 accuracy (shown in Table 3.4), estimating respectively the chance of the top-1 or one of top-10 (from beam search) matching the golden. Since there are often many good responses to a input, top-10 accuracy appears to be closer to the real world setting.

As shown in Table 3.4, COPYNET significantly outperforms RNNsearch, especially on DS-II. It suggests that introducing the copying mechanism helps the dialogue system master the patterns in dialogue and correctly identify the correct parts of input, often proper nouns, to replicate in the response. Since the filled substrings have no overlaps in DS-II, the performance of RNNSearch drops significantly as it cannot handle words unseen in training data. In contrast, the performance of COPYNET only drops slightly as it has learned to fill the slots with the copying mechanism and relies less on the representation of the words.

Case Study

As indicated by the examples in Figure 3.5, COPYNET accurately replicates the critical segments from the input with the copy-mode, and generates the rest of answers smoothly through the generate-mode. Note that in (2) and (3), the decoding sequence is not exactly the same with the standard one, yet still correct regarding to their meanings. In contrast, although RNNSearch usually generates answers in the right formats, it fails to catch the critical entities in all three cases because of the difficulty brought by the unseen words.

3.4 Related Work

Our work is partially inspired by the recent work of Pointer Networks (Vinyals et al., 2015a), in which a pointer mechanism (quite similar with the proposed copying mechanism) is used to predict the output sequence directly from the input. In addition to the difference with ours in application, (Vinyals et al., 2015a) cannot predict outside of the set of input sequence, while COPYNET can naturally combine generating and copying.

COPYNET is also related to the effort to solve the OOV problem in neural machine translation. Luong et al. (2015c) introduced a heuristics to post-process the translated sentence using annotations on the source sentence. In contrast COPYNET addresses the OOV problem in a more systemic way with an end-to-end model. However, as COPYNET copies the exact source words as the output, it cannot be directly applied to machine translation. However, such copying mechanism can be naturally extended to any types of references except for the input sequence, which will help in applications with heterogeneous source and target sequences such as machine translation.

The copying mechanism can also be viewed as carrying information over to the next stage without any nonlinear transformation. Similar ideas are proposed for training very deep neural networks in (Srivastava et al., 2015; He et al., 2015) for classification tasks, where shortcuts are built between layers for the direct carrying of information. Copying from the source sequence can also be seen as a special effort to utilise ”memory”, as we discussed as a hybrid representation of short-term memory in Seq2Seq learning framework, sharing some similarity with memory-based approaches like (Weston et al., 2014; Sukhbaatar et al., 2015; Graves et al., 2014)

3.5 Conclusion and Future Work

We proposed COPYNET to incorporate copying into the sequence-to-sequence learning framework. COPYNET can nicely integrate the regular way of word generation in the decoder with the new copying mechanism which can choose sub-sequences in the input sequence and put them at proper places in the output sequence. Our empirical study on both synthetic data sets and real world data sets demonstrates the efficacy of COPYNET.

Chapter 4

Search-Engine Guided Non-Parametric Neural Machine Translation

4.1 Introduction

Neural machine translation is a recently proposed paradigm in machine translation, where a single neural network, often consisting of encoder and decoder recurrent networks, is trained end-to-end to map from a source sentence to its corresponding translation(Bahdanau et al., 2014; Cho et al., 2014b; Sutskever et al., 2014; Kalchbrenner and Blunsom, 2013). The success of neural machine translation, which has already been adopted by major industry players in machine translation(Wu et al., 2016; Crego et al., 2016), is often attributed to the advances in building and training recurrent networks as well as the availability of large-scale parallel corpora for machine translation.

Neural machine translation is most characteristically distinguished from the existing approaches to machine translation, such as phrase-based statistical machine translation(Koehn et al., 2003), in that it projects a sequence of discrete source symbols into a continuous space and decodes back the corresponding translation. This allows one to easily incorporate other auxiliary information into the neural machine translation system as long as such auxiliary information could be encoded into a continuous space using a neural network. This property has been noticed recently and used for building more advanced translation systems such as multilingual translation (Firat et al., 2016a; Luong et al., 2015a), multi-source

translation (Zoph and Knight, 2016; Firat et al., 2016b), multimodal translation (Caglayan et al., 2016) and syntax guided translation (Nadejde et al., 2017; Eriguchi et al., 2017).

In this paper, we first notice that this ability in incorporating arbitrary meta-data by neural machine translation allows us to naturally extend it to a model in which a neural machine translation system explicitly takes into account a full training set consisting of source-target sentence pairs (in this paper we refer them as a general translation memory). We can build a neural machine translation system that considers not only a given source sentence, which is to be translated but also a set of training sentence pairs in the process of translation. To do so, we propose a novel extension of attention-based neural machine translation that seamlessly fuses two information streams, each of which corresponds to the current source sentence and a set of training sentence pairs, respectively.

A major technical challenge, other than designing such a neural machine translation system, is the scale of a training parallel corpus which often consists of hundreds of thousands to millions of sentence pairs. We address this issue by incorporating an off-the-shelf black-box search engine into the proposed neural machine translation system. The proposed approach first queries a search engine, which indexes a whole training set, with a given source sentence, and the proposed neural translation system translates the source sentence while incorporating all the retrieved training sentence pairs. In this way, the proposed translation system automatically adapts to the search engine and its ability to retrieve relevant sentence pairs from a training corpus.

We evaluate the proposed search engine guided neural machine translation (SEG-NMT) on three language pairs (En-Fr, En-De, and En-Es, in both directions) from JRC-Acquis Corpus(Steinberger et al., 2006) which consists of documents from a legal domain. This corpus was selected to demonstrate the efficacy of the proposed approach when a training corpus and a set of test sentences are both from a similar domain. Our experiments reveal that the proposed approach exploits the availability of the retrieved training sentence pairs very well, achieving significant improvement over the strong baseline of attention-based neural machine translation(Bahdanau et al., 2014).

4.2 Background

4.2.1 Neural Machine Translation

In this paper, we start from a recently proposed, and widely used, attention-based neural machine translation model(Bahdanau et al., 2014). The attention-based neural translation model is a conditional recurrent language model of a conditional distribution $p(Y|X)$ over all possible translations $Y = \{y_1, \ldots, y_T\}$ given a source sentence $X = \{x_1, \ldots, x_{T_x}\}$. This conditional recurrent language model is an autoregressive model that estimates the conditional probability as $p(Y|X) = \prod_{t=1}^{T} p(y_t|y_{<t}, X)$. Each term on the right hand side is approximated by a recurrent network by

$$p(y_t|y_{<t}, X) \propto \exp\left(g\left(y_t, z_t; \theta_g\right)\right), \tag{4.1}$$

where $z_t = f(z_{t-1}, y_{t-1}, c_t(X, z_{t-1}, y_{t-1}); \theta_f)$. g and f correspond to a read-out function that maps the hidden state z_t into a distribution over a target vocabulary, and a recurrent activation function that summarizes all the previously decoded target symbols $y_1, \ldots, y_{t-1}$ with respect to the time-dependent context vector $c_t(X; \theta_e)$, respectively. Both of these functions are parametrized, and their parameters are learned jointly to maximize the log-likelihood of a training parallel corpus. $c_t(X, z_{t-1}, y_{t-1})$ is composed of a bidirectional recurrent network encoder and an attention mechanism. The source sequence X is first encoded into a set of annotation vector $\{h_1, \ldots, h_{T_x}\}$, each of which is a concatenation of the hidden states of the forward and reverse recurrent networks. The attention mechanism, which is implemented as a feedforward network with a single hidden layer, then computes an attention score $\alpha_{t,\tau}$ for each hidden state h_τ given the previously decoded target symbol y_{t-1} and the previous decoder hidden state z_{t-1}:

$$\alpha_{t,\tau} = \frac{\exp\left\{\phi_{\text{att}}(h_\tau, y_{t-1}, z_{t-1})\right\}}{\sum_{\tau'=1}^{T_x} \exp\left\{\phi_{\text{att}}(h_{\tau'}, y_{t-1}, z_{t-1})\right\}}.$$

These attention scores are used to compute the time-dependent context vector c_t as

$$c_t = \sum_{\tau=1}^{T_x} \alpha_{t,\tau} h_\tau. \tag{4.2}$$

The attention-based neural machine translation system is end-to-end trained to maximize the likelihood of a correct translation given a corresponding source sentence. During testing, a given source sentence is translated by searching for the most likely translation from a trained model. The entire process of training and testing can be considered as compressing the whole training corpus into a neural machine translation system, as the training corpus is discarded once training is over.

4.2.2 Translation Memory

Translation memory is a computer-aided translation tool widely used by professional human translators. It is a database of pairs of source phrase and its translation. This database is constructed incrementally as a human translator translates sentences. When a new source sentence is present, a set of (overlapping) phrases from the original sentence are queried against the translation memory, and the corresponding entries are displayed to the human translator to speed up the process of translation. Due to the problem of sparsity (Sec.5.2 of (Cho, 2015)), exact matches rarely occur, and approximate string matching is often used.

In this paper, we consider a more general notion of translation memory in which not only translation phrase pairs but any kind of translation pairs are stored. In this more general definition, a training parallel corpus is also considered a translation memory. This saves us from building a phrase table(Koehn et al., 2003), which is yet another active research topic, but requires us to be efficient and flexible in retrieving relevant translation pairs given a source sentence, as the issue of data sparsity amplifies. This motivates us to come up with an efficient query algorithm tied together with a downstream translation model that can overcome the problem of data sparsity.

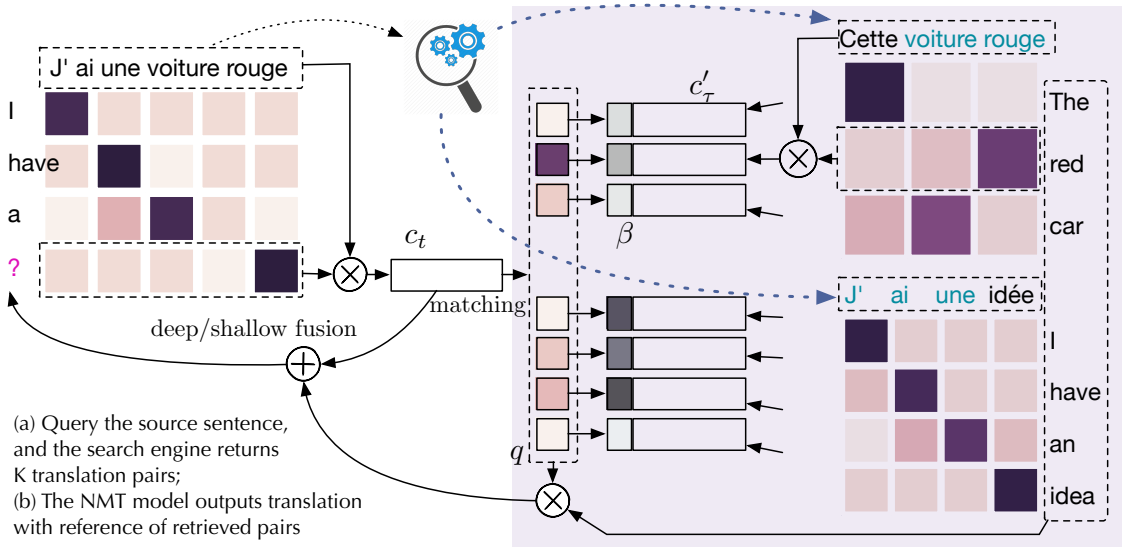


Figure 4.1: The overall architecture of the proposed SEG-NMT. The shaded box includes the module which handles a set of translation pairs retrieved in the first stage. The heat maps represent the attention scores between the source sentences (left-to-right) and the corresponding translations (top-to-down).

4.3 Search Engine Guided Non-Parametric Neural Machine Translation

We propose a non-parametric neural machine translation model guided by an off-the-shelf, efficient search engine. Unlike the conventional neural machine translation system, the proposed model does not discard a training corpus but maintain and actively exploit it in the test time. This effectively makes the proposed neural translation model a fully non-parametric model.

The proposed nonparametric neural translation model consists of two stages. The first stage is a retrieval stage, in which the proposed model queries a training corpus, or equivalently a translation memory, to retrieve a set of source-translation pairs given a current source sentence. To maximize the computational efficiency, first we utilize an off-the-shelf, highly-optimized search engine to quickly retrieve a large set of similar source sentences, and their translations, after which the top-K pairs are selected using approximate string

matching based on edit distance.

In the second stage, a given source sentence is translated by an attention-based neural machine translation model, which we refer to as a *search engine guided neural machine translation* (SEG-NMT), and incorporates the retrieved translation pairs from the first stage. In order to maximize the use of the retrieved pairs, we build a novel extension of the attention-based model that performs attention not only over the source symbols but also over the retrieved symbols (and their respective translations). We further allow the model an option to copy over a target symbol directly from the retrieved translation pairs. The overall architecture with a simple translation example of the proposed SEG-NMT is shown in Fig. 4.1 for reference.

4.3.1 Retrieval Stage

We refer to the first stage as a *retrieval stage*. In this stage, we go over the entire training set $\mathcal{M} = \{(X^n, Y^n)\}_{n=1}^N$ to find pairs whose source side is similar to a current source X. That is, we define a similarity function $s(X, X')$, and find (X^n, Y^n) where $s(X, X^n)$ is large.

Similarity score function s In this paper, we constrain ourselves to a setting in which only a neural translation model is trainable. That is, we do not assume the availability of other trainable sentence similarity functions. This allows us to focus entirely on the effectiveness of the proposed algorithm while being agnostic to the choice of similarity metric. Under this constraint, we follow an earlier work by (Li et al., 2016b) and use a fuzzy matching score which is defined as

$$s_{\text{fuzzy}}(X, X') = 1 - \frac{D_{\text{edit}}(X, X')}{\max\left(|X|, |X'|\right)}, \tag{4.3}$$

where D_{edit} is an edit distance.

Off-the-shelf Search Engine The computational complexity of the similarity search grows linearly with the size of the translation memory which in our case contains all the pairs from a training corpus. Despite the simplicity and computational efficiency of the similarity score in Eq. (4.3), this is clearly not practical, as the size of the training corpus is often in

Algorithm 1 Greedy selection procedure to maximize the coverage of the source symbols.

Require: input X, translation memory $\mathcal{M}$

Obtain the subset $\tilde{M} \subseteq M$ using an off-the-shelf search engine;
Re-rank retrieved pairs $(X', Y') \in \tilde{M}$ using the similarity score function s in descending order;
Initialize the dictionary of selected pairs $R = \emptyset$;
Initialize the coverage score $c = 0$;
for $k = 1...|\tilde{M}|$ **do**
    $c_{\text{tmp}} = \sum_{x \in X} \delta\left[x \in R.\text{keys} \cup \{X'_k\}\right] / |X|$
    if $c_{\text{tmp}} > c$ **then**
        $c = c_{\text{tmp}}$; $R \leftarrow \{X'_k : Y'_k\}$
return R

the order of hundreds of thousands or even tens of millions. We overcome this issue of scalability by incorporating an off-the-shelf search engine, more specifically Apache Lucene.[1] We then use Lucene to retrieve an initial set of translation pairs based on the source side, and use the similarity score above to re-rank them.

Final selection process Let $\tilde{\mathcal{M}} \in \mathcal{M}$ be an initial set of translation pairs returned by Lucene. We rank the translation pairs within this set by $s(X, X')$. We design and test two methods for selecting the final set from this initial set based on the similarity scores. The first method is a top-K retrieval, where we simply return the K most similar translation pairs from $\tilde{\mathcal{M}}$. The second method returns an adaptive number of translation pairs based on the coverage of the symbols x in the current source sentence X within the retrieved translation pairs. We select greedily starting from the most similar translation pair, as described in Alg. 4.

4.3.2 Translation Stage

In the second stage, we build a novel extension of the attention-based neural machine translation, SEG-NMT, that seamlessly fuses both a current source sentence and a set $\hat{M}$ of retrieved translation pairs. In a high level, the proposed SEG-NMT first stores each target symbol of each retrieved translation pair into a key-value memory(Miller et al., 2016). At each time step of the decoder, SEG-NMT first performs attention over the current source

[1] https://lucene.apache.org/core/

sentence to compute the time-dependent context vector based on which the key-value memory is queried. SEG-NMT fuses information from both context vector of the current source sentence and the retrieved value from the key-value memory to generate a next symbol.

Key-Value Memory For each retrieved translation pair $(X', Y') \in \hat{\mathcal{M}}$, we run a full attention-based neural machine translation model,[2] specified by a parameter set θ, and obtain, for each target symbol $y'_t \in Y'$, a decoder's hidden state z'_t and an associated time-dependent context vector c'_t (see Eq. (4.2) which summarizes a subset of the source sentence X' that best describes y'_t). We consider c'_t as a key and (z'_t, y'_t) as a value, and store all of them from all the retrieved translation pairs in a key-value memory. Note that this approach is agnostic to how many translation pairs were retrieved during the first stage.

Matching and Retrieval At each time step of the SEG-NMT decoder, we first compute the context vector c_t given the previous decoder hidden state z_t, the previously decoded symbol y_{t-1} and all the annotation vector h_τ's, as in Eq. (4.2). This context vector is used as a key for querying the key-value memory. Instead of hard matching, we propose *soft matching* based on a bilinear function, where we compute the matching score of each key-value slot by

$$q_{t,\tau} = \frac{\exp\{E(c_t, c'_\tau)\}}{\sum_{\tau'} \exp\{E(c_t, c'_{\tau'})\}}. \tag{4.4}$$

where $E(c_t, c'_\tau) = c_t^T M c'_\tau$ and M is a trainable matrix.

These scores are used to retrieve a value from the key-value memory. In the case of the decoder's hidden states, we retrieve a weighted sum: $\tilde{z}_t = \sum_\tau q_{t,\tau} z'_\tau$; In the case of target symbols, we consider each computed score as a probability of the corresponding target symbol. That is, $p_{\text{copy}}(y'_\tau) = q_{t,\tau}$, similarly to the pointer network(Vinyals et al., 2015a).

Incorporation We consider two separate approaches to incorporating the retrieved values from the key-value memory, motivated by (Gulcehre et al., 2015). The first approach, called *deep fusion*, weighted-average the retrieved hidden state $\tilde{z}_t$ and the decoder's hidden state

[2] We use a single copy of attention-based model for both key extraction and translation.

z_t:

$$z_{\text{fusion}} = \zeta_t \cdot \tilde{z}_t + (1 - \zeta_t) \cdot z_t \tag{4.5}$$

when computing the output distribution $p(y_t|y_{<t}, X, \mathcal{M})$ (see Eq. (4.1)). The second approach is called *shallow fusion* and computes the output distribution as a mixture:

$$\begin{aligned} p(y_t|y_{<t}, X, \mathcal{M}) = \zeta_t p_{\text{copy}}(y_t) \\ +(1 - \zeta_t) p(y_t|y_{<t}, X). \end{aligned} \tag{4.6}$$

This is equivalent to copying over a retrieved target symbol y'_τ with the probability of $\zeta_t p_{\text{copy}}(y_\tau)$ as the next target symbol(Gulcehre et al., 2016; Gu et al., 2016a).

In both of the approaches, there is a gating variable ζ_t. As each target symbol may require a different source of information, we let this variable be determined automatically by the proposed SEG-NMT. That is, we introduce another feedforward network that computes $\zeta_t = f_{\text{gate}}(c_t, z_t, \tilde{z}_t)$. This gate closes when the retrieved pairs are not useful for predicting the next target symbol y_t, and opens otherwise.

Algorithm 2 Learning for SEG-NMT

Require: Search engine F_{SS}, MT model θ, SEG model θ', M, λ, η, parallel training set $\mathcal{D}$, translation memory $\mathcal{M}$.

Initialize $\phi = \{\theta, \theta', M, \lambda, \eta\}$;
Set the number of returned answers as K;
while stopping criterion is not met **do**
    Draw a translation pair: $(X, Y) \sim \mathcal{D}$;
    Obtain memory pairs $\{X'_k, Y'_k\}_{k=1}^{K} = F_{SS}(X, \mathcal{M})$
    Reference Memory $C = \emptyset$.
    for $k = 1...K$ **do** # generate dynamic keys
        Let $Y'_k = \{y'_1, ..., y'_{T'}\}, X'_k = \{x'_1, ..., x'_{T'_s}\}$
        for $\tau = 1...T'$ **do**
            Generate key $c'_\tau = f_{\text{att}}(y'_{<\tau}, X'_k)$
            Initialize coverage $\beta_\tau = 0$.
            $C \leftarrow (c'_\tau, y'_\tau, \beta_\tau)$
    Let $Y = \{y_1, ..., y_T\}, X = \{x_1, ..., x_{T_s}\}$
    for $t = 1...T$ **do** # translate each word
        Generate query $c_t = f_{\text{att}}(y_{<t}, X)$
        for $\tau = 1...T'$ **do** Read $c'_\tau, y'_\tau, \beta_\tau \in C$
            Compute the score $q_{t,\tau}$ using Eq. 4.7;
            Gompute the gate ζ_t with f_{gate};
            Update $\beta_\tau \leftarrow \beta_\tau + q_{t,\tau} \cdot \zeta_t$;
        Compute the probability $p(y_t|\cdot)$
            –option1: shallow-fusion, Eq. 4.6
            –option2: deep-fusion, Eq. 4.5
    Update $\phi \leftarrow \phi + \gamma \frac{\partial}{\partial \phi} \sum_{t=1}^{T} \log p(y_t|\cdot)$

Coverage In the preliminary experiments, we notice that the access pattern of the key-value memory was highly skewed toward only a small number of slots. Motivated by the coverage penalty from (Tu et al., 2016b), we propose to augment the bilinear matching function (in Eq. (4.4)) with a coverage vector $\beta_{t,\tau}$ such that

$$E(c_t, c'_\tau) = c_t^T M c'_\tau - \lambda \beta_{t-1,\tau}, \tag{4.7}$$

where the coverage vector is defined as $\beta_{t,\tau} = \sum_{t'=1}^{t} q_{t',\tau} \cdot \zeta_{t'}$. λ is a trainable parameter.

4.3.3 Learning and Inference

The proposed model, including both the first and second stages, can be trained end-to-end to maximize the log-likelihood given a parallel corpus. For practical training, we preprocess a training parallel corpus by augmenting each sentence pair with a set of translation pairs retrieved by a search engine, while ensuring that the exact copy is not included in the retrieved set. See Alg. 5 for a detailed description. During testing, we search through the whole training set to retrieve relevant translation pairs. Similarly to a standard neural translation model, we use beam search to decode the best translation given a source sentence.

4.4 Related Work

The principal idea of SEG-NMT shares major similarities with the example-based machine translation (EBMT) (Zhang and Vogel, 2005; Callison-Burch et al., 2005; Phillips, 2012) which indexes parallel corpora with suffix arrays and retrieves translations on the fly at test time. However, to the best of our knowledge, SEG-NMT is the first work incorporating any attention-based neural machine translation architectures and can be trained end-to-end efficiently, showing superior performance and scalability compared to the conventional statistical EBMT.

SEG-NMT has also been largely motivated by recently proposed multilingual attention-based neural machine translation models(Firat et al., 2016a; Zoph and Knight, 2016). Similar to these multilingual models, our model takes into account more information than a current source sentence. This allows the model to better cope with any uncertainty or ambiguity arising from a single source sentence. More recently, this kind of larger context translation has been applied to cross-sentential modeling, where the translation of a current sentence is done with respect to previous sentences(Jean et al., 2017; Wang et al., 2017).

(Devlin et al., 2015) proposed an automatic image caption generation model based on

nearest neighbours. In their approach, a given image is queried against a set of training pairs of images and their corresponding captions. They then proposed to use a median caption among those nearest neighboring captions, as a generated caption of the given image. This approach shares some similarity with the first stage of the proposed SEG-NMT. However, unlike their approach, we *learn to generate* a sentence rather than simply choose one among retrieved ones.

(Bordes et al., 2015) proposed a memory network for large-scale simple question-answering using an entire Freebase(Bollacker et al., 2008). The output module of the memory network used simple n-gram matching to create a small set of candidate facts from the Freebase. Each of these candidates was scored by the memory network to create a representation used by the response module. This is similar to our approach in that it exploits a black-box search module (n-gram matching) for generating a small candidate set.

A similar approach was very recently proposed for deep reinforcement learning by (Pritzel et al., 2017), where they store pairs of observed state and the corresponding (estimated) value in a key-value memory to build a non-parametric deep Q network. We consider it as a confirmation of the general applicability of the proposed approach to a wider array of problems in machine learning. In the context of neural machine translation, (Kaiser et al., 2017b) also proposed to use an external key-value memory to remember training examples in the test time. Due to the lack of efficient search mechanism, they do not update the memory jointly with the translation model, unlike the proposed approach in this paper.

One important property of the proposed SEG-NMT is that it relies on an external, black-box search engine to retrieve relevant translation pairs. Such a search engine is used both during training and testing, and an obvious next step is to allow the proposed SEG-NMT to more intelligently query the search engine, for instance, by reformulating a given source sentence. Recently, (Nogueira and Cho, 2017) proposed task-oriented query reformulation in which a neural network is trained to use a black-box search engine to maximize the recall of relevant documents, which can be integrated into the proposed SEG-NMT. We leave this as future work.

4.5 Experimental Settings

Dataset	En-Fr	En-De	En-Es
# Train Pairs	744,528	717,096	697,187
# Dev Pairs	2,665	2,454	2,533
# Test Pairs	2,655	2,483	2,596
# En/sent.	29.44	33.43	32.10
# Other/sent.	33.34	33.44	34.95

Table 4.1: Statistics from the JRC-Acquis corpus. We use BPE subword symbols.

Data We use the JRC-Acquis corpus(Steinberger et al., 2006) for evaluating the proposed SEG-NMT model.[3] The JRC-Acquis corpus consists of the total body of European Union (EU) law applicable to the member states. The text in this corpus is well structured, and most of the text in this corpus are related, making it an ideal test bed to evaluate the proposed SEG-NMT which relies on the availability of appropriate translation pairs from a training set. This corpus was also used by (Li et al., 2016b) in investigating the combination of translation memory and phrase-based statistical machine translation, making it suitable for our proposed method to evaluate on.

We select three language pairs, namely, En-Fr, En-Es, and En-De, for evaluation. For each language pair, we uniformly select 3000 sentence pairs at random for both the development and test sets. The rest is used as a training set, after removing any sentence which contains special characters only. We use sentences of lengths up to 80 and 100 from the training and dev/test sets respectively. We do not lowercase the text, and use byte-pair encoding (BPE)(Sennrich et al., 2015b) to extract a vocabulary of 20,000 subword symbols. See Table 4.1 for detailed statistics.

Retrieval Stage We use Apache Lucene to index a whole training set and retrieve 100 pairs per source sentence for the initial retrieval. These 100 pairs are scored against the current source sentence using the fuzzy matching score from Eq. (4.3) to select top-K relevant translation pairs. We vary K among 1 and 2 during training and among $1, 2, 4, 8, 16$ during testing to investigate the trade-off between retrieval and translation quality. During testing, we also evaluate the effect of adaptively deciding the number of retrieved pairs using the proposed greedy selection algorithm (Alg. 4).

[3] http://optima.jrc.it/Acquis/JRC-Acquis.3.0/corpus/

		En-Fr		En-De		En-Es	
		→	←	→	←	→	←
Dev	TM	46.62	42.53	34.99	42.45	40.84	39.71
	NMT	58.95	59.69	44.94	50.20	50.54	55.02
	Copy	60.34	61.61	-	-	-	-
	Ours	**64.16**	**64.64**	**49.26**	**55.63**	**57.62**	**60.28**
Test	TM	46.64	43.17	34.61	41.83	39.55	37.73
	NMT	59.42	60.11	43.98	49.74	50.48	54.66
	Copy	60.55	62.02	-	-	-	-
	Ours	**64.60**	**65.11**	**48.80**	**55.33**	**57.27**	**59.34**

Table 4.2: The BLEU scores on JRC-Acquis corpus.

Translation Stage We use a standard attention-based neural machine translation model(Bahdanau et al., 2014) with 1,024 gated recurrent units(GRU)(Cho et al., 2014b) on each of the encoder and decoder. We train both the vanilla model as well as the proposed SEG-NMT based on this configuration from scratch using Adam(Kingma and Ba, 2014) with the initial learning rate set to 0.001. We use a minibatch of up to 32 sentence pairs. We early-stop based on the development set performance. For evaluation, we use beam search with width set to 5.

In the case of the proposed SEG-NMT, we parametrize the metric matrix M in the similarity score function from Eq. (4.7) to be diagonal and initialized to an identity matrix. λ in Eq. (4.7) is initialized to 0. The gating network f_{gate} is a feedforward network with a single hidden layer, just like the attention mechanism f_{att}. We use either deep fusion or shallow fusion in our experiments.

4.6 Result and Analysis

In Table 8.1, we present the BLEU scores obtained on all the three language pairs (both directions each) using three approaches; TM – a carbon copy of the target side of a retrieved translation pair with the highest matching score, NMT - a baseline translation model, and our proposed SEG-NMT model. It is evident from the table that the proposed SEG-NMT significantly outperforms the baseline model in all the cases, and that this improvement

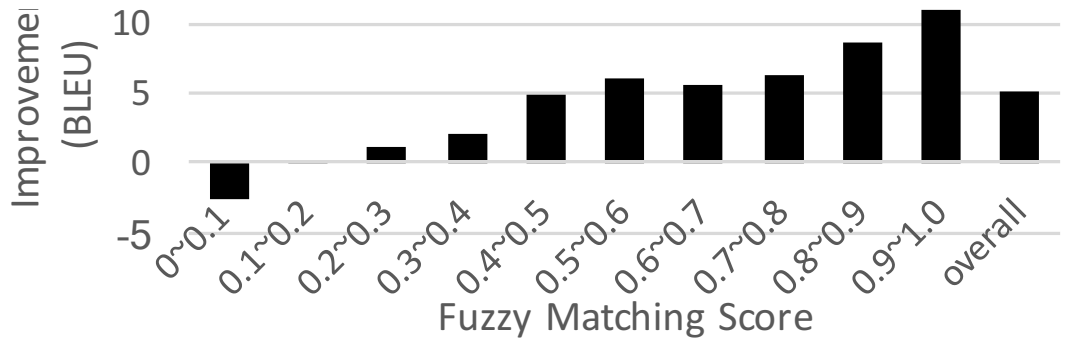


Figure 4.2: The improvement over the baseline by SEG-NMT on Fr→En w.r.t. the fuzzy matching scores of one retrieved translation pair.

is not merely due to their copying over the most similar translation from a training set. For Fr-En and En-Fr, we also present the performance of using a "CopyNet" (Gu et al., 2016a) variant which uses a copying mechanism directly over the target side of the searched translation pair. This CopyNet variant helps but not as much as the proposed approach. We conjecture this happens because our proposal of using a key-value memory captures the relationship between the source and target tokens in the retrieved pairs more tightly.

Fuzzy matching score v.s. Quality For Fr→En, we broke down the development set into a set of bins according to the matching score of a retrieved translation pair, and computed the BLEU score for each bin. As shown in Fig. 4.2, we note that the improvement grows as the relevance of the retrieved translation pair increases. This verifies that SEG-NMT effectively exploits retrieved translation pairs, but also suggests a future improvement for the case in which no relevant translation pair exists in a training set.

Effect of the # of Retrieved Translation Pairs Once the proposed model is trained, it can be used with a varying number of retrieved translation pairs. We test the model trained on Fr→En with different numbers of retrieved translation pairs, and present the BLEU scores in Fig. 4.3. We notice that the translation quality increases as the number of retrieved pairs increase up to approximately four, but from there on it degrades. We believe this happens as the retrieved sentences become less related to the current source sentence. The best quality was achieved when the proposed greedy selection algorithm in Alg. 4 was used, in which case 4.814 translation pairs were retrieved on average.

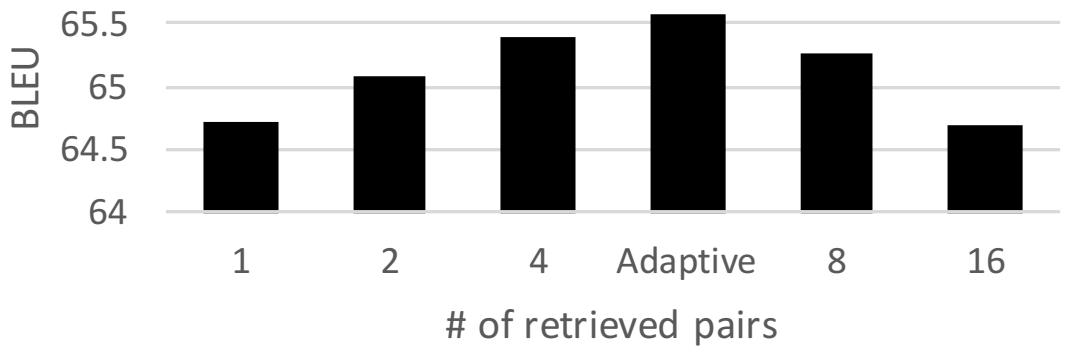


Figure 4.3: The BLEU scores on Fr→En using varying numbers of retrieved translation pairs during testing. The model was trained once. "Adaptive" refers to the proposed greedy selection in Alg. 4.

Deep vs. Shallow Fusion On both directions of En-Fr, we implemented and tested both deep and shallow fusion (Eqs. (4.5)–(4.6)) for incorporating the information from the retrieved translation pairs. With deep fusion only, the BLEU scores on the development set improved over the baseline by 1.30 and 1.20 respectively, while the improvements were 5.21 and 4.95, respectively. This suggests that the proposed model effectively exploits the availability of target symbols in the retrieved translation pairs. All other experiments were thus done using shallow fusion only.

Examples We list two good examples and one in which the proposed method makes a mistake, in Fig. 4.4. From these examples, we see that the proposed SEG-NMT selects a term or phrase used in a retrieved pair whenever there are ambiguities or multiple correct translations. For instance, in the first example, SEG-NMT translated "précis" into "exact" which was used in the retrieved pair, while the baseline model chose "precise". A similar behavior is found with "examen" in the second example. This behavior helps the proposed SEG-NMT generate a translation of which style and choice of vocabulary match better with translations from a training corpus, which improves the overall consistency of the translation.

Efficiency In general, there are two points at which computational complexity increases. The first point occurs at the retrieval stage which incurs almost no overhead as we rely on an efficient search engine (which retrieves a pair within several milliseconds.) In the

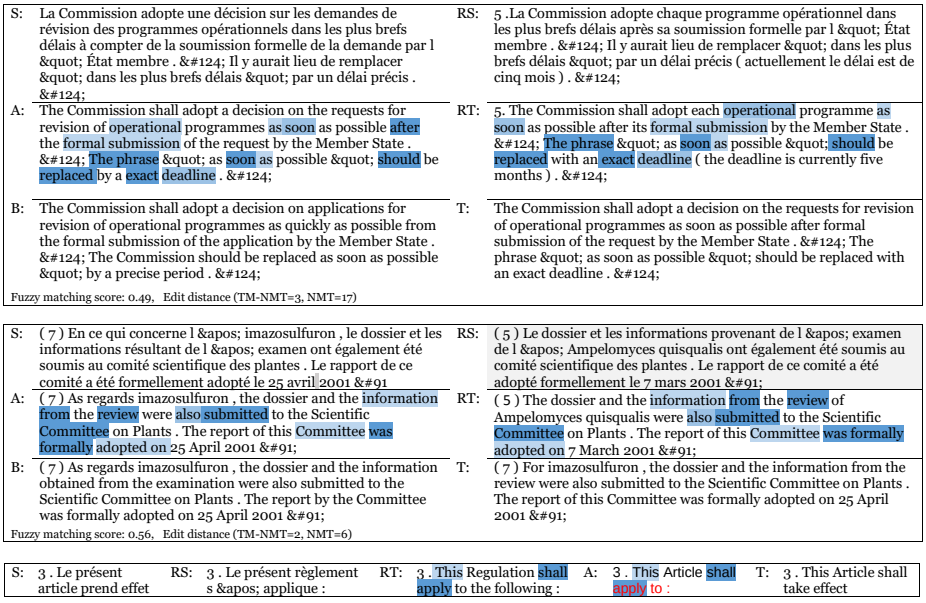


Figure 4.4: Three examples from the Fr→En test set. For the proposed SEG-NMT model, one translation pair is retrieved from the training set. Each token in the translation by the proposed approach and its corresponded token (if it exists) in the retrieved pair are shaded in blue according to the gating variable ζ_t from Eq. (4.6). In all, we show: (S) the source sentence. (RS) the source side of a retrieved pair. (RT) the target side of the retrieved pair. (A) the translation by the proposed approach. (B) the translation by the baseline. (T) the reference translation.

translation stage, the complexity of indexing the key-value memory grows w.r.t. the # of tokens in the retrieved pairs. This increase is however constant with a reasonably-set max # of retrieved pairs. Note that the memory can be pre-populated for all the training pairs.

4.7 Conclusion

We proposed a practical, non-parametric extension of attention-based neural machine translation by utilizing an off-the-shelf, black-box search engine for quickly selecting a small subset of training translation pairs. The proposed model, called SEG-NMT, then learns to incorporate both the source- and target-side information from these retrieved pairs to improve the translation quality. We empirically showed the effectiveness of the proposed approach on the JRC-Acquis corpus using six language pair-directions.

Although the proposed approach is in the context of machine translation, it is generally applicable to a wide array of problems. By embedding an input of any modality into a fixed vector space and using approximate search(Johnson et al., 2017a), this approach can, for instance, be used for open-domain question answering, where the seamless fusion of multiple sources of information retrieved by a search engine is at the core. We leave these as future work.

Chapter 5

Universal Neural Machine Translation for Extremely Low Resource Languages

5.1 Introduction

Neural Machine Translation (NMT) (Bahdanau et al., 2014) has achieved remarkable translation quality in various on-line large-scale systems (Wu et al., 2016; Devlin, 2017) as well as achieving state-of-the-art results on Chinese-English translation (Hassan et al., 2018). With such large systems, NMT showed that it can scale up to immense amounts of parallel data in the order of tens of millions of sentences. However, such data is not widely available for all language pairs and domains. In this paper, we propose a novel universal multi-lingual NMT approach focusing mainly on low resource languages to overcome the limitations of NMT and leverage the capabilities of multi-lingual NMT in such scenarios.

Our approach utilizes multi-lingual neural translation system to share lexical and sentence level representations across multiple source languages into one target language. In this setup, some of the source languages may be of extremely limited or even zero data. The lexical sharing is represented by a universal word-level representation where various words from all source languages share the same underlaying representation. The sharing module utilizes monolingual embeddings along with seed parallel data from all languages to build the universal representation. The sentence-level sharing is represented by a model

of language experts which enables low-resource languages to utilize the sentence representation of the higher resource languages. This allows the system to translate from any language even with tiny amount of parallel resources.

We evaluate the proposed approach on 3 different languages with tiny or even zero parallel data. We show that for the simulated "zero-resource" settings, our model can consistently outperform a strong multi-lingual NMT baseline with a tiny amount of parallel sentence pairs.

5.2 Motivation

Neural Machine Translation (NMT) (Bahdanau et al., 2014; Sutskever et al., 2014) is based on Sequence-to-Sequence encoder-decoder model along with an attention mechanism to enable better handling of longer sentences (Bahdanau et al., 2014). Attentional sequence-to-sequence models are modeling the log conditional probability of the translation Y given an input sequence X. In general, the NMT system θ consists of two components: an encoder θ_e which transforms the input sequence into an array of continuous representations, and a decoder θ_d that dynamically reads the encoder's output with an attention mechanism and predicts the distribution of each target word. Generally, θ is trained to maximize the likelihood on a training set consisting of N parallel sentences:

$$\begin{aligned}\mathcal{L}(\theta) &= \frac{1}{N}\sum_{n=1}^{N}\log p\left(Y^{(n)}|X^{(n)};\theta\right) \\ &= \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}\log p\left(y_t^{(n)}|y_{1:t-1}^{(n)}, f_t^{\text{att}}(h_{1:T_s}^{(n)})\right)\end{aligned} \tag{5.1}$$

where at each step, f_t^{att} builds the attention mechanism over the encoder's output $h_{1:T_s}$. More precisely, let the vocabulary size of source words as V

$$h_{1:T_s} = f^{\text{ext}}\left[e_{x_1}, ..., e_{x_{T_s}}\right], \quad e_x = E^I(x) \tag{5.2}$$

Figure 5.1: BLEU scores reported on the test set for Ro-En. The amount of training data effects the translation performance dramatically using a single NMT model.

where $E^I \in \mathbb{R}^{V \times d}$ is a look-up table of source embeddings, assigning each individual word a unique embedding vector; f^{ext} is a sentence-level feature extractor and is usually implemented by a multi-layer bidirectional RNN (Bahdanau et al., 2014; Wu et al., 2016), recent efforts also achieved the state-of-the-art using non-recurrence f^{ext}, e.g. ConvS2S (Gehring et al., 2017) and Transformer (Vaswani et al., 2017).

Extremely Low-Resource NMT Both θ_e and θ_d should be trained to converge using parallel training examples. However, the performance is highly correlated to the amount of training data. As shown in Figure. 5.1, the system cannot achieve reasonable translation quality when the number of the parallel examples is extremely small ($N \approx 13k$ sentences, or not available at all $N = 0$).

Multi-lingual NMT Lee et al. (2016) and Johnson et al. (2017b) have shown that NMT is quite efficient for multilingual machine translation. Assuming the translation from K source languages into one target language, a system is trained with maximum likelihood

on the mixed parallel pairs $\{X^{(n,k)}, Y^{(n,k)}\}_{k=1...K}^{n=1...N_k}$, that is

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{k=1}^{K} \sum_{n=1}^{N_k} \log p\left(Y^{(n,k)} | X^{(n,k)}; \theta\right) \tag{5.3}$$

where $N = \sum_{k=1}^{K} N_k$. As the input layer, the system assumes a multilingual vocabulary which is usually the union of all source language vocabularies with a total size as $V = \sum_{k=1}^{K} V_k$. In practice, it is essential to shuffle the multilingual sentence pairs into mini-batches so that different languages can be trained equally. Multi-lingual NMT is quite appealing for low-resource languages; several papers highlighted the characteristic that make it a good fit for that such as Lee et al. (2016), Johnson et al. (2017b), Zoph et al. (2016) and Firat et al. (2016a). Multi-lingual NMT utilizes the training examples of multiple languages to regularize the models avoiding over-fitting to the limited data of the smaller languages. Moreover, the model transfers the translation knowledge from high-resource languages to low-resource ones. Finlay, the decoder part of the model is sufficiently trained since it shares multilingual examples from all languages.

5.2.1 Challenges

Despite the success of training multi-lingual NMT systems; there are a couple of challenges to leverage them for zero-resource languages:

Lexical-level Sharing Conventionally, a multi-lingual NMT model has a vocabulary that represents the union of the vocabularies of all source languages. Therefore, the multi-lingual words do not practically share the same embedding space since each word has its own representation. This does not pose a problem for languages with sufficiently large amount of data, yet it is a major limitation for extremely low resource languages since most of the vocabulary items will not have enough, if any, training examples to get a reliably trained models.

A possible solution is to share the surface form of all source languages through sharing sub-units such as subwords (Sennrich et al., 2015b) or characters (Kim et al., 2016; Luong and Manning, 2016; Lee et al., 2016). However, for an arbitrary low-resource language

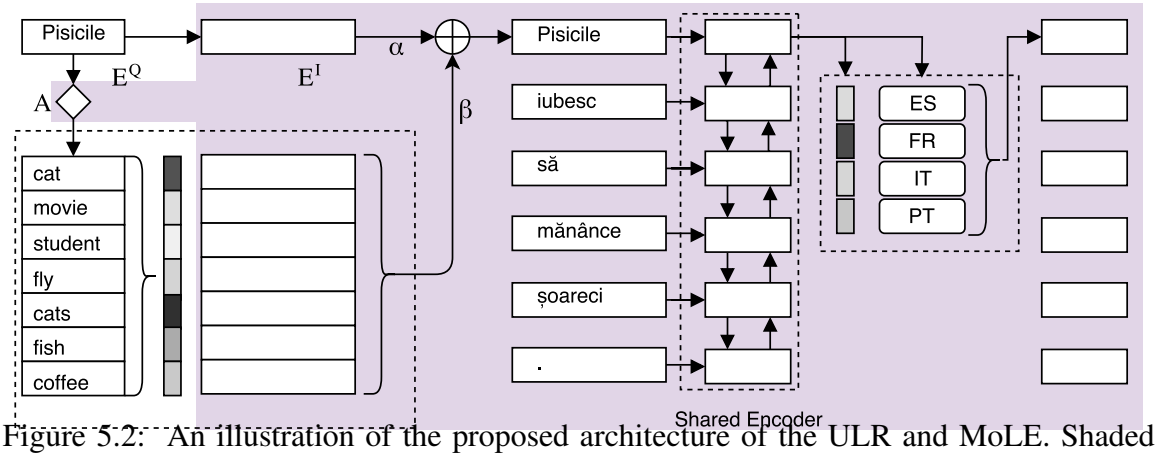


Figure 5.2: An illustration of the proposed architecture of the ULR and MoLE. Shaded parts are trained within NMT model while unshaded parts are not changed during training.

we cannot assume significant overlap in the lexical surface forms compared to the high-resource languages. The low-resource language may not even share the same character set as any high-resource language. It is crucial to create a shared semantic representation across all languages that does not rely on surface form overlap.

Sentence-level Sharing It is also crucial for low-resource languages to share source sentence representation with other similar languages. For example, if a language shares syntactic order with another language it should be feasible for the low-resource language to share such representation with another high recourse language. It is also important to utilize monolingual data to learn such representation since the low or zero resource language may have monolingual resources only.

5.3 Universal Neural Machine Translation

We propose a Universal NMT system that is focused on the scenario where minimal parallel sentences are available. As shown in Fig. 5.2, we introduce two components to extend the conventional multi-lingual NMT system (Johnson et al., 2017b): Universal Lexical Representation (ULR) and Mixture of Language Experts (MoLE) to enable both word-level and sentence-level sharing, respectively.

5.3.1 Universal Lexical Representation (ULR)

As we highlighted above, it is not straightforward to have a universal representation for all languages. One potential approach is to use a shared source vocabulary, but this is not adequate since it assumes significant surface-form overlap in order being able to generalize between high-resource and low-resource languages. Alternatively, we could train monolingual embeddings in a shared space and use these as the input to our MT system. However, since these embeddings are trained on a monolingual objective, they will not be optimal for an NMT objective. If we simply allow them to change during NMT training, then this will not generalize to the low-resource language where many of the words are unseen in the parallel data. Therefore, our goal is to create a shared embedding space which (a) is trained towards NMT rather than a monolingual objective, (b) is not based on lexical surface forms, and (c) will generalize from the high-resource languages to the low-resource language.

We propose a novel representation for multi-lingual embedding where each word from any language is represented as a probabilistic mixture of universal-space word embeddings. In this way, semantically similar words from different languages will naturally have similar representations. Our method achieves this utilizing a discrete (but probabilistic) "universal token space", and then learning the embedding matrix for these universal tokens directly in our NMT training.

Lexicon Mapping to the Universal Token Space We first define a discrete universal token set of size M into which all source languages will be projected. In principle, this could correspond to any human or symbolic language, but all experiments here use English as the basis for the universal token space. As shown in Figure 5.2, we have multiple embedding representations. E^Q is language-specific embedding trained on monolingual data and E^K is universal tokens embedding. The matrices E^K and E^Q are created beforehand and are not trainable during NMT training. E^U is the embedding matrix for these universal tokens which is learned during our NMT training. It is worth noting that shaded parts in Figure5.2 are trainable during NMT training process. Therefore, each source word e_x is represented as a mixture of universal tokens M of E^U.

$$e_x = \sum_{i=1}^{M} E^U\left(u_i\right) \cdot q(u_i|x) \tag{5.4}$$

where E^U is an NMT embedding matrix, which is learned during NMT training.

The mapping q projects the multilingual words into the universal space based on their semantic similarity. That is, $q(u|x)$ is a distribution based on the distance $D_s(u, x)$ between u and x as:

$$q(u_i|x) = \frac{e^{D(u_i,x)/\tau}}{\sum_{u_j} e^{D(u_j,x)/\tau}} \tag{5.5}$$

where τ is a temperature and $D(u_i, x)$ is a scalar score which represents the similarity between source word x and universal token u_i:

$$D(u, x) = E^K(u) \cdot A \cdot E^Q(x)^T \tag{5.6}$$

where $E^K(u)$ is the "key" embedding of word u, $E^Q(x)$ is the "query" embedding of source word x. The transformation matrix A, which is initialized to the identity matrix, is learned during NMT training and shared across all languages.

This is a key-value representation, where the queries are the monolingual language-specific embedding, the keys are the universal tokens embeddings and the values are a probabilistic distribution over the universal NMT embeddings. This can represent unlimited multi-lingual vocabulary that has never been observed in the parallel training data. It is worth noting that the trainable transformation matrix A is added to the query matching mechanism with the main purpose to tune the similarity scores towards the translation task. A is shared across all languages and optimized discriminatively during NMT training such that the system can fine-tune the similarity score $q()$ to be optimal for NMT.

Shared Monolingual Embeddings In general, we create one E^Q matrix per source language, as well as a single E^K matrix in our universal token language. For Equation 5.6 to make sense and generalize across language pairs, all of these embedding matrices must live in a similar semantic space. To do this, we first train off-the-shelf monolingual word embeddings in each language, and then learn one projection matrix per source language which maps the original monolingual embeddings into E^K space. Typically, we need a list

of *source word - universal token* pairs (seeds S_k) to train the projection matrix for language k. Since vectors are normalized, learning the optimal projection is equivalent to finding an orthogonal transformation O_k that makes the projected word vectors as close as to its corresponded universal tokens:

$$\begin{aligned} &\max_{O_k} \sum_{(\tilde{x},\tilde{y})\in S_k} \left(E^{Q_k}(\tilde{x}) \cdot O_k\right) \cdot E^K(\tilde{y})^T \\ &\text{s.t. } O_k^T O_k = I, \quad k = 1, ..., K \end{aligned} \tag{5.7}$$

which can be solved by SVD decomposition based on the seeds (Smith et al., 2017). In this paper, we chose to use a short list of seeds from automatic word-alignment of parallel sentences to learn the projection. However, recent efforts (Artetxe et al., 2017a; Conneau et al., 2018) also showed that it is possible to learn the transformation without any seeds, which makes it feasible for our proposed method to be utilized in purely zero parallel resource cases.

It is worth noting that O_k is a language-specific matrix which maps the monolingual embeddings of each source language into a similar semantic space as the universal token language.

Interpolated Embeddings Certain lexical categories (e.g. function words) are poorly captured by Equation 5.4. Luckily, function words often have very high frequency, and can be estimated robustly from even a tiny amount of data. This motivates an interpolated e_x where embeddings for very frequent words are optimized directly and not through the universal tokens:

$$\alpha(x)E^I(x) + \beta(x)\sum_{i=1}^{M} E^U\left(u_i\right) \cdot q(u_i|x) \tag{5.8}$$

Where $E^I(x)$ is a language-specific embedding of word x which is optimized during NMT training. In general, we set $\alpha(x)$ to 1.0 for the top k most frequent words in each language, and 0.0 otherwise, where k is set to 500 in this work. It is worth noting that we do not use an absolute frequency cutoff because this would cause a mismatch between high-resource and low-resource languages, which we want to avoid. We keep $\beta(x)$ fixed to 1.0.

An Example To give a concrete example, imagine that our target language is English (En), our high-resource auxiliary source languages are Spanish (Es) and French (Fr), and our low-resource source language is Romanian (Ro). En is also used for the universal token set. We assume to have 10M+ parallel Es-En and Fr-En, and a few thousand in Ro-En. We also have millions of monolingual sentences in each language.

We first train word2vec embeddings on monolingual corpora from each of the four languages. We next align the Es-En, Fr-En, and Ro-En parallel corpora and extract a seed dictionary of a few hundred words per language, e.g., `gato` $\rightarrow$ `cat`, `chien` $\rightarrow$ `dog`. We then learn three matrices O_1, O_2, O_3 to project the Es, Fr and Ro embeddings ($E^{Q_1}, E^{Q_2}, E^{Q_3}$), into En ($E^K$) based on these seed dictionaries. At this point, Equation 5.5 should produce *reasonable* alignments between the source languages and En, e.g., $q(\texttt{horse}|\texttt{magar}) = 0.5$, $q(\texttt{donkey}|\texttt{magar}) = 0.3$, $q(\texttt{cow}|\texttt{magar}) = 0.2$, where `magar` is the Ro word for `donkey`.

5.3.2 Mixture of Language Experts (MoLE)

As we paved the road for having a universal embedding representation; it is crucial to have a language-sensitive module for the encoder that would help in modeling various language structures which may vary between different languages. We propose a Mixture of Language Experts (MoLE) to model the sentence-level universal encoder. As shown in Fig. 5.2, an additional module of mixture of experts is used after the last layer of the encoder. Similar to (Shazeer et al., 2017), we have a set of expert networks and a gating network to control the weight of each expert. More precisely, we have a set of expert networks as $f_1(h), ..., f_K(h)$ where for each expert, a two-layer feed-forward network which reads the output hidden states h of the encoder is utilized. The output of the MoLE module h' will be a weighted sum of these experts to replace the encoder's representation:

$$h' = \sum_{k=1}^{K} f_k(h) \cdot \text{softmax}(g(h))_k, \tag{5.9}$$

where an one-layer feed-forward network $g(h)$ is used as a gate to compute scores for all the experts.

In our case, we create one expert per auxiliary language. In other words, we train to

only use expert f_i when training on a parallel sentence from auxiliary language i. Assume the language $1...K-1$ are the auxiliary languages. That is, we have a multi-task objective as:

$$\mathcal{L}^{\text{gate}} = \sum_{k=1}^{K-1} \sum_{n=1}^{N_k} \log \left[\text{softmax}\left(g(h)\right)_k\right] \tag{5.10}$$

We do not update the MoLE module for training on a sentence from the low-resource language. Intuitively, this allows us to represent each token in the low-resource language as a context-dependent mixture of the auxiliary language experts.

5.4 Experiments

We extensively study the effectiveness of the proposed methods by evaluating on three "almost-zero-resource" language pairs with variant auxiliary languages. The vanilla single-source NMT and the multi-lingual NMT models are used as baselines.

5.4.1 Settings

Dataset We empirically evaluate the proposed Universal NMT system on 3 languages – Romanian (Ro) / Latvian (Lv) / Korean (Ko) – translating to English (En) in near zero-resource settings. To achieve this, single or multiple auxiliary languages from Czech (Cs), German (De), Greek (El), Spanish (Es), Finnish (Fi), French (Fr), Italian (It), Portuguese (Pt) and Russian (Ru) are jointly trained. The detailed statistics and sources of the available parallel resource can be found in Table 5.1, where we further down-sample the corpora for the targeted languages to simulate zero-resource.

It also requires additional large amount of monolingual data to obtain the word embeddings for each language, where we use the latest Wikipedia dumps [5] for all the languages.

[1] http://www.statmt.org/wmt16/translation-task.html
[2] https://sites.google.com/site/koreanparalleldata/
[3] http://www.statmt.org/europarl/
[4] http://opus.lingfil.uu.se/MultiUN.php (subset)
[5] https://dumps.wikimedia.org/

	Zero-Resource Translation			Auxiliary High-Resource Translation								
source	Ro	Ko	Lv	Cs	De	El	Es	Fi	Fr	It	Pt	Ru
corpora	WMT16[1]	KPD[2]	Europarl v8[3]									UN [4]
size	612k	97k	638k	645k	1.91m	1.23m	1.96m	1.92m	2.00m	1.90m	1.96m	11.7m
subset	0/6k/60k	10k	6k	/								2.00m

Table 5.1: Statistics of the available parallel resource in our experiments. All the languages are translated to English.

Typically, the monolingual corpora are much larger than the parallel corpora. For validation and testing, the standard validation and testing sets are utilized for each targeted language.

Preprocessing All the data (parallel and monolingual) have been tokenized and segmented into subword symbols using byte-pair encoding (BPE) (Sennrich et al., 2015b). We use sentences of length up to 50 subword symbols for all languages. For each language, a maximum number of $40,000$ BPE operations are learned and applied to restrict the size of the vocabulary. We concatenate the vocabularies of all source languages in the multilingual setting where special a "language marker " have been appended to each word so that there will be no embedding sharing on the surface form. Thus, we avoid sharing the representation of words that have similar surface forms though with different meaning in various languages.

Architecture We implement an attention-based neural machine translation model which consists of a one-layer bidirectional RNN encoder and a two-layer attention-based RNN decoder. All RNNs have 512 LSTM units (Hochreiter and Schmidhuber, 1997). Both the dimensions of the source and target embedding vectors are set to 512. The dimensionality of universal embeddings is also the same. For a fair comparison, the same architecture is also utilized for training both the vanilla and multilingual NMT systems. For multilingual experiments, $1 \sim 5$ auxiliary languages are used. When training with the universal tokens, the temperature τ (in Eq. 5.6) is fixed to 0.05 for all the experiments.

Learning All the models are trained to maximize the log-likelihood using Adam (Kingma and Ba, 2014) optimizer for 1 million steps on the mixed dataset with a batch size of 128.

The dropout rates for both the encoder and the decoder is set to 0.4. We have open-sourced an implementation of the proposed model. [6]

5.4.2 Back-Translation

We utilize back-translation (BT) (Sennrich et al., 2016) to encourage the model to use more information of the zero-resource languages. More concretely, we build the synthetic parallel corpus by translating on monolingual data[7] with a trained translation system and use it to train a backward direction translation model. Once trained, the same operation can be used on the forward direction. Generally, BT is difficult to apply for zero resource setting since it requires a reasonably good translation system to generate good quality synthetic parallel data. Such a system may not be feasible with tiny or zero parallel data. However, it is possible to start with a trained multi-NMT model.

5.4.3 Preliminary Experiments

Training Monolingual Embeddings We train the monolingual embeddings using `fastText`[8] (Bojanowski et al., 2017) over the Wikipedia corpora of all the languages. The vectors are set to 300 dimensions, trained using the default setting of skip-gram . All the vectors are normalized to norm 1.

Pre-projection In this paper, the pre-projection requires initial word alignments (seeds) between words of each source language and the universal tokens. More precisely, for the experiments of Ro/Ko/Lv-En, we use the target language (En) as the universal tokens; `fast_align`[9] is used to automatically collect the aligned words between the source languages and English.

Src	Aux	Multi	+ULR	+ MoLE
Ro	Cs De El Fi		18.02	18.37
	Cs De El Fr		19.48	19.52
	De El Fi It		19.11	19.33
	Es Fr It Pt	14.83	20.01	**20.51**
Lv	Es Fr It Pt	7.68	10.86	11.02
	Es Fr It Pt Ru	7.88	12.40	**13.16**
Ko	Es Fr It Pt	2.45	5.49	**6.14**

Table 5.2: Scores over variant source languages (6k sentences for Ro & Lv, and 10k for Ko). "Multi" means the Multi-lingual NMT baseline.

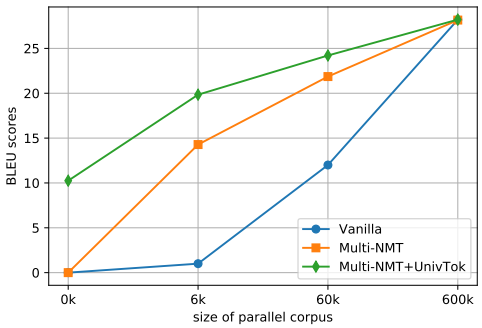


Figure 5.3: BLEU score vs corpus size

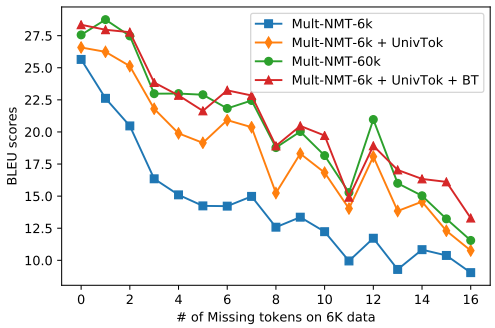


Figure 5.4: BLEU score vs unknown tokens

5.5 Results

We show our main results of multiple source languages to English with different auxiliary languages in Table 5.2. To have a fair comparison, we use only 6k sentences corpus for both Ro and Lv with all the settings and 10k for Ko. It is obvious that applying both the universal tokens and mixture of experts modules improve the overall translation quality for all the language pairs and the improvements are additive.

To examine the influence of auxiliary languages, we tested four sets of different combinations of auxiliary languages for Ro-En and two sets for Lv-En. It shows that Ro performs best when the auxiliary languages are all selected in the same family (Ro, Es, Fr, It and

[6]https://github.com/MultiPath/NA-NMT/tree/universal_translation
[7]We used News Crawl provided by WMT16 for Ro-En.
[8]https://github.com/facebookresearch/fastText
[9]https://github.com/clab/fast_align

Models	BLEU
Vanilla	1.21
Multi-NMT	14.94
Closest Uni-Token Only	5.83
Multi-NMT + ULR + (A=I)	18.61
Multi-NMT + ULR	**20.01**
Multi-NMT + BT	17.91
Multi-NMT + ULR + BT	**22.35**
Multi-NMT + ULR + MoLE	20.51
Multi-NMT + ULR + MoLE + BT	**22.92**
Full data (612k) NMT	**28.34**

Table 5.3: BLEU scores evaluated on test set (6k), compared with ULR and MoLE. "vanilla" is the standard NMT system trained only on Ro-En training set

Pt are all from the Romance family of European languages) which makes sense as more knowledge can be shared across the same family. Similarly, for the experiment of Lv-En, improvements are also observed when adding Ru as additional auxiliary language as Lv and Ru share many similarities because of the geo-graphical influence even though they don't share the same alphabet.

We also tested a set of Ko-En experiments to examine the generalization capability of our approach on non-European languages while using languages of Romance family as auxiliary languages. Although the BLEU score is relatively low, the proposed methods can consistently help translating less-related low-resource languages. It is more reasonable to have similar languages as auxiliary languages.

5.5.1 Ablation Study

We perform thorough experiments to examine effectiveness of the proposed method; we do ablation study on Ro-En where all the models are trained based on the same Ro-En corpus with 6k sentences.

As shown in Table 5.3, it is obvious that 6k sentences of parallel corpora completely fails to train a vanilla NMT model. Using Multi-NMT with the assistance of 7.8M auxiliary language sentence pairs, Ro-En translation performance gets a substantial improvement which, however, is still limited to be usable. By contrast, the proposed ULR boosts the

(a)	Source	situatia este putin diferita atunci cand sunt analizate separat raspunsurile barbatilor si ale femeilor .
	Reference	the situation is slightly different when responses are analysed separately for men and women .
	Mul-6k	the situation is less different when it comes to issues of men and women .
	Mul-60k	the situation is at least different when it is weighed up separately by men and women .
	Lex-6k	the situation is somewhat different when we have a separate analysis of women 's and women 's responses .
	Lex-6k +BT	the situation is slightly different when it is analysed separately from the responses of men and women .

(b)	Source	ce nu stim este in cat timp se va intampla si cat va dura .
	Reference	what we don ' t know is how long all of that will take and how long it will last .
	Lex (Romance)	what we do not know is how long it will be and how long it will take .
	Lex (Non-Rom)	what we know is as long as it will happen and how it will go

(c)	Source	limita de greutate pentru acestea dateaza din anii ' 80 , cand air india a inceput sa foloseasca grafice cu greutatea si inaltimea ideale .
	Reference	he weight limit for them dates from the ' 80s , when air india began using ideal weight and height graphics .
	Lex (A = I)	the weight limit for these dates back from the 1960s , when the chinese air began to use physiars with weight and the right height .
	Lex	the weight limit for these dates dates from the 1980s , when air india began to use the standard of its standard and height .

Figure 5.5: Three sets of examples on Ro-En translation with variant settings.

Multi-NMT significantly with +5.07 BLEU, which is further boosted to +7.98 BLEU when incorporating sentence-level information using both MoLE and BT. Furthermore, it is also shown that ULR works better when a trainable transformation matrix A is used (4th vs 5th row in the table). Note that, although still $5 \sim 6$ BLEU scores lower than the full data ($\times 100$ large) model.

We also measure the translation quality of simply training the vanilla system while replacing each token of the Ro sentence with its closet universal token in the projected embedding space, considering we are using the target languages (En) as the universal tokens. Although the performance is much worse than the baseline Multi-NMT, it still outperforms the vanilla model which implies the effectiveness of the embedding alignments.

Monolingual Data In Table. 5.3, we also showed the performance when incorporating the monolingual Ro corpora to help the UniNMT training in both cases with and without ULR. The back-translation improves in both cases, while the ULR still obtains the best score which indicates that the gains achieved are additive.

Corpus Size As shown in Fig. 5.3, we also evaluated our methods with varied sizes – 0k[10], 6k, 60k and 600k – of the Ro-En corpus. The vanilla NMT and the multi-lingual NMT are used as baselines. It is clear in all cases that the performance gets better when the training corpus is larger. However, the multilingual with ULR works much better with

[10]For 0k experiments, we used the pre-projection learned from 6k data. It is also possible to use unsupervised learned dictionary.

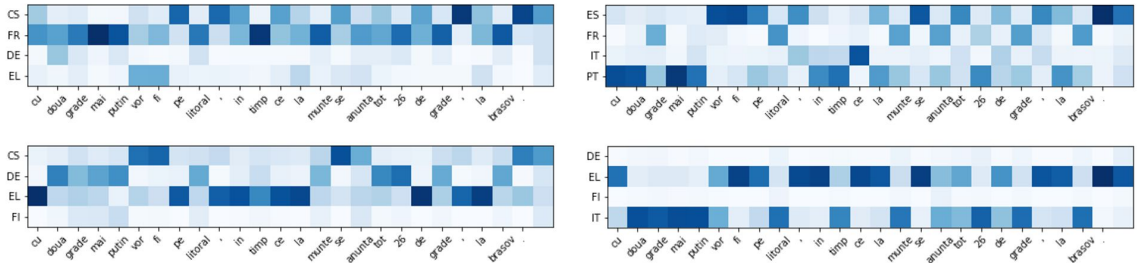


Figure 5.6: The activation visualization of mixture of language experts module on one randomly selected Ro source sentences trained together with different auxiliary languages. Darker color means higher activation score.

a small amount of training examples. Note that, the usage of ULR universal tokens also enables us to directly work on a "pure zero" resource translation with a shared multilingual NMT model.

Unknown Tokens One explanation on how ULR help the translation for almost zero resource languages is it greatly cancel out the effects of missing tokens that would cause out-of-vocabularies during testing. As in Fig. 5.4, the translation performance heavily drops when it has more "unknown" which cannot be found in the given 6k training set, especially for the typical multilingual NMT. Instead, these "unknown" tokens will naturally have their embeddings based on ULR projected universal tokens even if we never saw them in the training set. When we apply back-translation over the monolingual data, the performance further improves which can almost catch up with the model trained with 60k data.

5.5.2 Qualitative Analysis

Examples Figure 5.5 shows some cherry-picked examples for Ro-En. Example (a) shows how the lexical selection get enriched when introducing ULR (Lex-6K) as well as when adding Back Translation (Lex-6K-BT). Example (b) shows the effect of using romance vs non-romance languages as the supporting languages for Ro. Example (c) shows the importance of having a trainable A as have been discussed; without trainable A the model confuses "india" and "china" as they may have close representation in the mono-lingual embeddings.

Visualization of MoLE Figure 5.6 shows the activations along with the same source sentence with various auxiliary languages. It is clear that MoLE is effectively switching between the experts when dealing with zero-resource language words. For this particular example of Ro, we can see that the system is utilizing various auxiliary languages based on their relatedness to the source language. We can approximately rank the relatedness based of the influence of each language. For instance, the influence can be approximately ranked as $Es \approx Pt > Fr \approx It > Cs \approx El > De > Fi$, which is interestingly close to the grammatical relatedness of Ro to these languages. On the other hand, Cs has a strong influence although it does not fall in the same language family with Ro, we think this is due to the geo-graphical influence between the two languages since Cs and Ro share similar phrases and expressions. This shows that MoLE learns to utilize resources from similar languages.

5.5.3 Fine-tuning a Pre-trained Model

All the described experiments above had the low resource languages jointly trained with all the auxiliary high-resource languages, where the training of the large amount of high-resource languages can be seen as a sort of regularization. It is also common to train a model on high-resource languages first, and then fine-tune the model on a small resource language similar to transfer learning approaches (Zoph et al., 2016). However, it is not trivial to effectively fine-tune NMT models on extremely low resource data since the models easily over-fit due to over-parameterization of the neural networks.

In this experiment, we have explored the fine-tuning tasks using our approach. First, we train a Multi-NMT model (with ULR) on {Es, Fr, It, Pt}-En languages only to create a zero-shot setting for Ro-En translation. Then, we start fine-tuning the model with $6k$ parallel corpora of Ro-En, with and without ULR. As shown in Fig. 5.7, both models improve a lot over the baseline. With the help of ULR, we can achieve a BLEU score of around 10.7 (also shown in Fig. 5.3) for Ro-En translation with "zero-resource" translation. The BLEU score can further improve to almost 20 BLEU after 3 epochs of training on $6k$ sentences using ULR. This is almost 6 BLEU higher than the best score of the baseline. It is worth noting that this fine-tuning is a very efficient process since it only takes less than 2 minutes

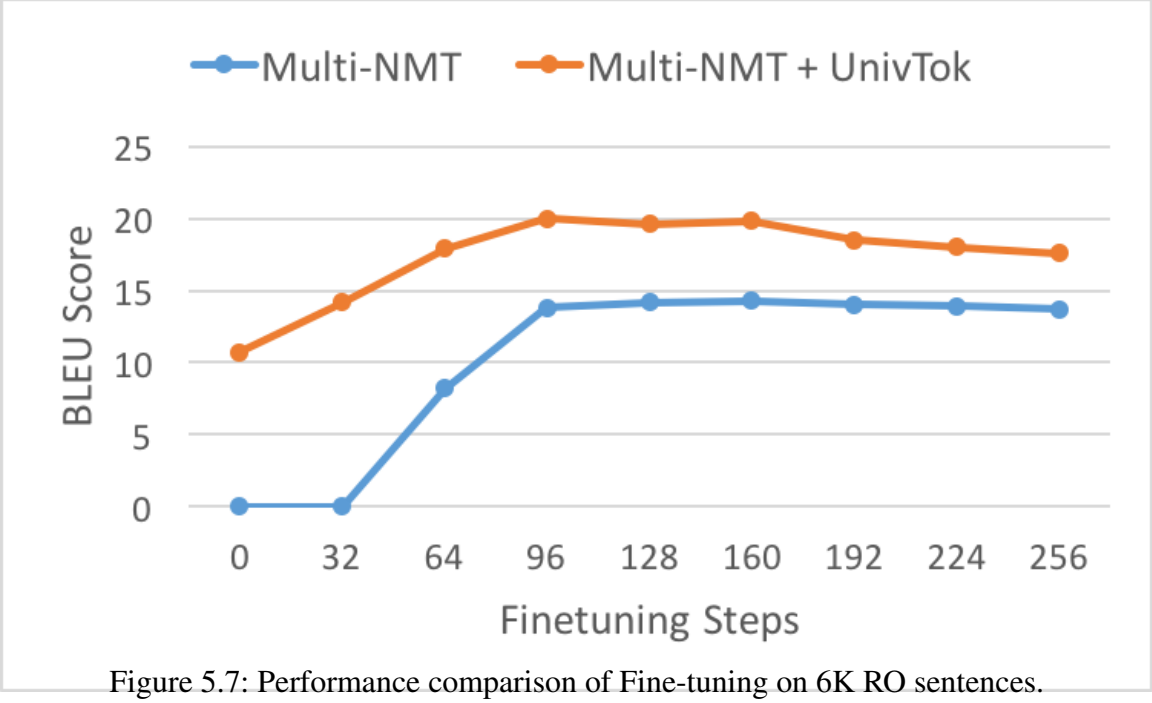


Figure 5.7: Performance comparison of Fine-tuning on 6K RO sentences.

to train for 3 epochs over such tiny amount of data. This is very appealing for practical applications where adapting a per-trained system on-line is a big advantage. As a future work, we will further investigate a better fine-tuning strategy such as meta-learning (Finn et al., 2017) using ULR.

5.6 Related Work

Multi-lingual NMT has been extensively studied in a number of papers such as Lee et al. (2016), Johnson et al. (2017b), Zoph et al. (2016) and Firat et al. (2016a). As we discussed, these approaches have significant limitations with zero-resource cases. Johnson et al. (2017b) is more closely related to our current approach, our work is extending it to overcome the limitations with very low-resource languages and enable sharing of lexical and sentence representation across multiple languages.

Two recent related works are targeting the same problem of minimally supervised or totally unsupervised NMT. Artetxe et al. (2018) proposed a totally unsupervised approach

depending on multi-lingual embedding similar to ours and dual-learning and reconstruction techniques to train the model from mono-lingual data only. Lample et al. (2018) also proposed a quite similar approach while utilizing adversarial learning.

5.7 Conclusion

In this paper, we propose a new universal machine translation approach that enables sharing resources between high resource languages and extremely low resource languages. Our approach is able to achieve 23 BLEU on Romanian-English WMT2016 using a tiny parallel corpus of 6k sentences, compared to the 18 BLEU of strong multi-lingual baseline system.

Chapter 6

Meta Learning for Low Resource Neural Machine Translation

6.1 Introduction

Despite the massive success brought by neural machine translation (NMT, Sutskever et al., 2014; Bahdanau et al., 2014; Vaswani et al., 2017), it has been noticed that the vanilla NMT often lags behind conventional machine translation systems, such as statistical phrase-based translation systems (PBMT, Koehn et al., 2003), for low-resource language pairs (see, e.g., Koehn and Knowles, 2017). In the past few years, various approaches have been proposed to address this issue. The first attempts at tackling this problem exploited the availability of monolingual corpora (Gulcehre et al., 2015; Sennrich et al., 2015a; Zhang and Zong, 2016). It was later followed by approaches based on multilingual translation, in which the goal was to exploit knowledge from high-resource language pairs by training a single NMT system on a mix of high-resource and low-resource language pairs (Firat et al., 2016a,b; Lee et al., 2016; Johnson et al., 2017b; Ha et al., 2016b). Its variant, transfer learning, was also proposed by Zoph et al. (2016), in which an NMT system is pretrained on a high-resource language pair before being finetuned on a target low-resource language pair.

In this paper, we follow up on these latest approaches based on multilingual NMT and propose a meta-learning algorithm for low-resource neural machine translation. We start by arguing that the recently proposed model-agnostic meta-learning algorithm (MAML,

Finn et al., 2017) could be applied to low-resource machine translation by viewing language pairs as separate tasks. This view enables us to use MAML to find the initialization of model parameters that facilitate fast adaptation for a new language pair with a minimal amount of training examples (§6.3). Furthermore, the vanilla MAML however cannot handle tasks with mismatched input and output. We overcome this limitation by incorporating the universal lexical representation (Gu et al., 2018b) and adapting it for the meta-learning scenario (§6.3.3).

We extensively evaluate the effectiveness and generalizing ability of the proposed meta-learning algorithm on low-resource neural machine translation. We utilize 17 languages from Europarl and Russian from WMT as the source tasks and test the meta-learned parameter initialization against five target languages (Ro, Lv, Fi, Tr and Ko), in all cases translating to English. Our experiments using only up to 160k tokens in each of the target task reveal that the proposed meta-learning approach outperforms the multilingual translation approach across all the target language pairs, and the gap grows as the number of training examples decreases.

6.2 Background

Neural Machine Translation (NMT) Given a source sentence $X = \{x_1, ..., x_{T'}\}$, a neural machine translation model factors the distribution over possible output sentences $Y = \{y_1, ..., y_T\}$ into a chain of conditional probabilities with a left-to-right causal structure:

$$p(Y|X;\theta) = \prod_{t=1}^{T+1} p(y_t|y_{0:t-1}, x_{1:T'};\theta), \tag{6.1}$$

where special tokens y_0 ($\langle \text{bos} \rangle$) and y_{T+1} ($\langle \text{eos} \rangle$) are used to represent the beginning and the end of a target sentence. These conditional probabilities are parameterized using a neural network. Typically, an encoder-decoder architecture (Sutskever et al., 2014; Cho et al., 2014a; Bahdanau et al., 2014) with a RNN-based decoder is used. More recently, architectures without any recurrent structures (Gehring et al., 2017; Vaswani et al., 2017) have been proposed and shown to speedup training while achieving state-of-the-art performance.

Low Resource Translation NMT is known to easily over-fit and result in an inferior performance when the training data is limited (Koehn and Knowles, 2017). In general, there are two ways for handling the problem of low resource translation: (1) utilizing the resource of unlabeled monolingual data, and (2) sharing the knowledge between low- and high-resource language pairs. Many research efforts have been spent on incorporating the monolingual corpora into machine translation, such as multi-task learning (Gulcehre et al., 2015; Zhang and Zong, 2016), back-translation (Sennrich et al., 2015a), dual learning (He et al., 2016) and unsupervised machine translation with monolingual corpora only for both sides (Artetxe et al., 2018; Lample et al., 2018; Yang et al., 2018).

For the second approach, prior researches have worked on methods to exploit the knowledge of auxiliary translations, or even auxiliary tasks. For instance, Cheng et al. (2016); Chen et al. (2017); Lee et al. (2017); Chen et al. (2018) investigate the use of a pivot to build a translation path between two languages even without any directed resource. The pivot can be a third language or even an image in multimodal domains. When pivots are not easy to obtain, Firat et al. (2016a); Lee et al. (2016); Johnson et al. (2017b) have shown that the structure of NMT is suitable for multilingual machine translation. Gu et al. (2018b) also showed that such a multilingual NMT system could improve the performance of low resource translation by using a universal lexical representation to share embedding information across languages. All the previous work for multilingual NMT assume the joint training of multiple high-resource languages naturally results in a universal space (for both the input representation and the model) which, however, is not necessarily true, especially for very low resource cases.

Meta Learning In the machine learning community, meta-learning, or learning-to-learn, has recently received interests. Meta-learning tries to solve the problem of "fast adaptation on new training data." One of the most successful applications of meta-learning has been on few-shot (or one-shot) learning (Lake et al., 2015), where a neural network is trained to readily learn to classify inputs based on only one or a few training examples. There are two categories of meta-learning:

1. learning a meta-policy for updating model parameters (see, e.g., Andrychowicz et al., 2016; Ha et al., 2016a; Mishra et al., 2017)

2. learning a good parameter initialization for fast adaptation (see, e.g., Finn et al., 2017; Vinyals et al., 2016; Snell et al., 2017).

In this paper, we propose to use a meta-learning algorithm for low-resource neural machine translation based on the second category. More specifically, we extend the idea of model-agnostic meta-learning (MAML, Finn et al., 2017) in the multilingual scenario.

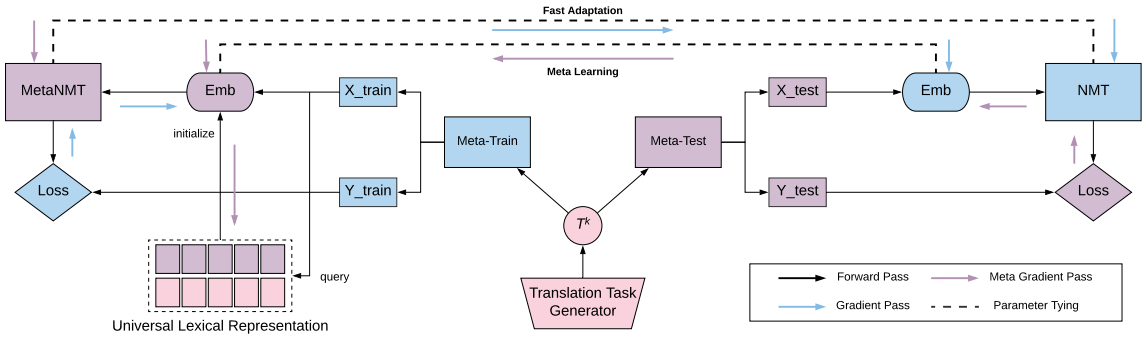


Figure 6.1: The graphical illustration of the training process of the proposed MetaNMT. For each episode, one task (language pair) is sampled for meta-learning. The boxes and arrows in blue are mainly involved in language-specific learning (§6.3.1), and those in purple in meta-learning (§6.3.2).

6.3 Meta Learning for Low-Resource Neural Machine Translation

The underlying idea of MAML is to use a set of source tasks $\{\mathcal{T}^1, \ldots, \mathcal{T}^K\}$ to find the initialization of parameters θ^0 from which learning a target task $\mathcal{T}^0$ would require only a small number of training examples. In the context of machine translation, this amounts to using many high-resource language pairs to find good initial parameters and training a new translation model on a low-resource language starting from the found initial parameters. This process can be understood as

$$\theta^* = \text{Learn}(\mathcal{T}^0; \text{MetaLearn}(\mathcal{T}^1, \ldots, \mathcal{T}^K)).$$

That is, we *meta-learn* the initialization from auxiliary tasks and continue to *learn* the target task. We refer the proposed meta-learning method for NMT to MetaNMT. See Fig. 6.1 for the overall illustration.

6.3.1 Learn: language-specific learning

Given any initial parameters θ^0 (which can be either random or meta-learned), the prior distribution of the parameters of a desired NMT model can be defined as an isotropic Guassian:

$$\theta_i \sim \mathcal{N}(\theta_i^0, 1/\beta),$$

where $1/\beta$ is a variance. With this prior distribution, we formulate the language-specific learning process Learn$(D_{\mathcal{T}}; \theta^0)$ as maximizing the log-posterior of the model parameters given data $D_{\mathcal{T}}$:

$$\begin{aligned}\text{Learn}(D_{\mathcal{T}}; \theta^0) &= \arg\max_{\theta} \mathcal{L}^{D_{\mathcal{T}}}(\theta) \\ &= \arg\max_{\theta} \sum_{(X,Y)\in D_{\mathcal{T}}} \log p(Y|X,\theta) - \beta\|\theta - \theta^0\|^2,\end{aligned}$$

where we assume $p(X|\theta)$ to be uniform. The first term above corresponds to the maximum likelihood criterion often used for training a usual NMT system. The second term discourages the newly learned model from deviating too much from the initial parameters, alleviating the issue of over-fitting when there is not enough training data. In practice, we solve the problem above by maximizing the first term with gradient-based optimization and early-stopping after only a few update steps. Thus, in the low-resource scenario, finding a good initialization θ^0 strongly correlates the final performance of the resulting model.

6.3.2 MetaLearn

We find the initialization θ^0 by repeatedly simulating low-resource translation scenarios using auxiliary, high-resource language pairs. Following Finn et al. (2017) we achieve this

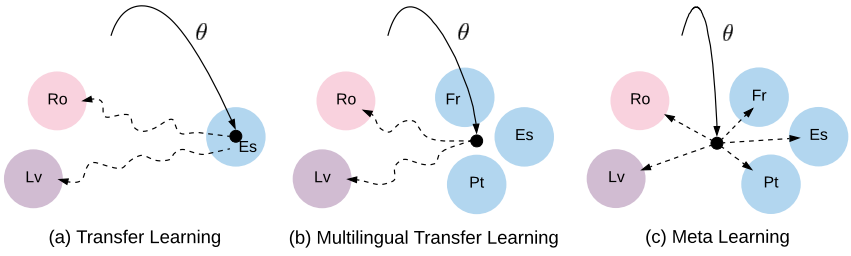


Figure 6.2: An intuitive illustration in which we use solid lines to represent the learning of initialization, and dashed lines to show the path of fine-tuning.

goal by defining the meta-objective function as

$$\mathcal{L}(\theta) = \mathbb{E}_k \mathbb{E}_{D_{\mathcal{T}^k}, D'_{\mathcal{T}^k}} \left[\sum_{(X,Y) \in D'_{\mathcal{T}^k}} \log p(Y|X; \text{Learn}(D_{\mathcal{T}^k}; \theta)) \right], \tag{6.2}$$

where $k \sim \mathcal{U}(\{1, \ldots, K\})$ refers to one meta-learning episode, and $D_{\mathcal{T}}$, $D'_{\mathcal{T}}$ follow the uniform distribution over $\mathcal{T}$'s data.

We maximize the meta-objective function using stochastic approximation (Robbins and Monro, 1951) with gradient descent. For each episode, we uniformly sample one source task at random, $\mathcal{T}^k$. We then sample two subsets of training examples independently from the chosen task, $D_{\mathcal{T}^k}$ and $D'_{\mathcal{T}^k}$. We use the former to *simulate* language-specific learning and the latter to *evaluate* its outcome. Assuming a single gradient step is taken only the with learning rate η, the simulation is:

$$\theta'_k = \text{Learn}(D_{\mathcal{T}^k}; \theta) = \theta - \eta \nabla_\theta \mathcal{L}^{D_{\mathcal{T}^k}}(\theta).$$

Once the simulation of learning is done, we evaluate the updated parameters θ'_k on $D'_{\mathcal{T}^k}$, The gradient computed from this evaluation, which we refer to as *meta-gradient*, is used to update the meta model θ. It is possible to aggregate multiple episodes of source tasks before updating θ:

$$\theta \leftarrow \theta - \eta' \sum_k \nabla_\theta \mathcal{L}^{D'_{\mathcal{T}^k}}(\theta'_k),$$

where η' is the meta learning rate.

Unlike a usual learning scenario, the resulting model θ^0 from this meta-learning procedure is not necessarily a good model on its own. It is however a good starting point for training a good model using only a few steps of learning. In the context of machine translation, this procedure can be understood as finding the initialization of a neural machine translation system that could quickly adapt to a new language pair by simulating such a fast adaptation scenario using many high-resource language pairs.

Meta-Gradient We use the following approximation property

$$H(x)v \approx \frac{\nabla(x + \nu v) - \nabla(x)}{\nu}$$

to approximate the meta-gradient:[1]

$$\begin{aligned}
&\nabla_\theta \mathcal{L}^{D'}(\theta') = \nabla_{\theta'} \mathcal{L}^{D'}(\theta') \nabla_\theta (\theta - \eta \nabla_\theta \mathcal{L}^D(\theta)) \\
&= \nabla_{\theta'} \mathcal{L}^{D'}(\theta') - \eta \nabla_{\theta'} \mathcal{L}^{D'}(\theta') H_\theta(\mathcal{L}^D(\theta)) \\
&\approx \nabla_{\theta'} \mathcal{L}^{D'}(\theta') - \frac{\eta}{\nu} \left[\nabla_\theta \mathcal{L}^D(\theta) \Big|_{\hat{\theta}} - \nabla_\theta \mathcal{L}^D(\theta) \Big|_{\theta} \right],
\end{aligned}$$

where ν is a small constant and

$$\hat{\theta} = \theta + \nu \nabla_{\theta'} \mathcal{L}^{D'}(\theta').$$

In practice, we find that it is also possible to ignore the second-order term, ending up with the following simplified update rule:

$$\nabla_\theta \mathcal{L}^{D'}(\theta') \approx \nabla_{\theta'} \mathcal{L}^{D'}(\theta'). \tag{6.3}$$

Related Work: Multilingual Transfer Learning The proposed MetaNMT differs from the existing framework of multilingual translation (Lee et al., 2016; Johnson et al., 2017b; Gu et al., 2018b) or transfer learning (Zoph et al., 2016). The latter can be thought of as

[1] We omit the subscript k for simplicity.

solving the following problem:

$$\max_{\theta} \mathcal{L}^{\text{multi}}(\theta) = \mathbb{E}_k \left[\sum_{(X,Y)\in D_k} \log p(Y|X;\theta) \right],$$

where D_k is the training set of the k-th task, or language pair. The target low-resource language pair could either be a part of joint training or be trained separately starting from the solution θ^0 found from solving the above problem.

The major difference between the proposed MetaNMT and these multilingual transfer approaches is that the latter do not consider how learning happens with the target, low-resource language pair. The former explicitly incorporates the learning process within the framework by simulating it repeatedly in Eq. (6.2). As we will see later in the experiments, this results in a substantial gap in the final performance on the low-resource task.

Illustration In Fig. 6.2, we contrast transfer learning, multilingual learning and meta-learning using three source language pairs (Fr-En, Es-En and Pt-En) and two target pairs (Ro-En and Lv-En). Transfer learning trains an NMT system specifically for a source language pair (Es-En) and finetunes the system for each target language pair (Ro-En, Lv-En). Multilingual learning often trains a single NMT system that can handle many different language pairs (Fr-En, Pt-En, Es-En), which may or may not include the target pairs (Ro-En, Lv-En). If not, it finetunes the system for each target pair, similarly to transfer learning. Both of these however aim at directly solving the source tasks. On the other hand, meta-learning trains the NMT system to be *useful for fine-tuning* on various tasks including the source and target tasks. This is done by repeatedly simulating the learning process on low-resource languages using many high-resource language pairs (Fr-En, Pt-En, Es-En).

6.3.3 Unified Lexical Representation

I/O mismatch across language pairs One major challenge that limits applying meta-learning for low resource machine translation is that the approach outlined above assumes the input and output spaces are shared across all the source and target tasks. This, however, does not apply to machine translation in general due to the vocabulary mismatch across

different languages. In multilingual translation, this issue has been tackled by using a vocabulary of sub-words (Sennrich et al., 2015a) or characters (Lee et al., 2016) shared across multiple languages. This surface-level sharing is however limited, as it cannot be applied to languages exhibiting distinct orthography (e.g., Indo-Euroepan languages vs. Korean.)

Universal Lexical Representation (ULR) We tackle this issue by dynamically building a vocabulary specific to each language using a key-value memory network (Miller et al., 2016; Gulcehre et al., 2018), as was done successfully for low-resource machine translation recently by Gu et al. (2018b). We start with multilingual word embedding matrices $\epsilon^k_{\text{query}} \in \mathbb{R}^{|V_k| \times d}$ pretrained on large monolingual corpora, where V_k is the vocabulary of the k-th language. These embedding vectors can be obtained with small dictionaries of seed word pairs (Artetxe et al., 2017b; Smith et al., 2017) or in a fully unsupervised manner (Zhang et al., 2017; **?**). We take one of these languages k' to build universal lexical representation consisting of a universal embedding matrix $\epsilon_u \in \mathbb{R}^{M \times d}$ and a corresponding key matrix $\epsilon_{\text{key}} \in \mathbb{R}^{M \times d}$, where $M < |V'_k|$. Both $\epsilon^k_{\text{query}}$ and ϵ_{key} are fixed during meta-learning. We then compute the language-specific embedding of token x from the language k as the convex sum of the universal embedding vectors by

$$\epsilon^0[x] = \sum_{i=1}^{M} \alpha_i \epsilon_u[i],$$

where $\alpha_i \propto \exp\left\{-\frac{1}{\tau}\epsilon_{\text{key}}[i]^\top A \epsilon^k_{\text{query}}[x]\right\}$ and τ is set to 0.05. This approach allows us to handle languages with different vocabularies using a fixed number of shared parameters (ϵ_u, ϵ_{key} and A.)

Learning of ULR It is not desirable to update the universal embedding matrix ϵ_u when fine-tuning on a small corpus which contains a limited set of unique tokens in the target language, as it could adversely influence the other tokens' embedding vectors. We thus estimate the change to each embedding vector induced by language-specific learning by a separate parameter $\Delta\epsilon^k[x]$:

$$\epsilon^k[x] = \epsilon^0[x] + \Delta\epsilon^k[x].$$

	# of sents.	# of En tokens	Dev	Test
Ro-En	0.61 M	16.66 M	–	31.76
Lv-En	4.46 M	67.24 M	20.24	15.15
Fi-En	2.63 M	64.50 M	17.38	20.20
Tr-En	0.21 M	5.58 M	15.45	13.74
Ko-En	0.09 M	2.33 M	6.88	5.97

Table 6.1: Statistics of full datasets of the target language pairs. BLEU scores on the dev and test sets are reported from a supervised Transformer model with the same architecture.

During language-specific learning, the ULR $\epsilon^0[x]$ is held constant, while only $\Delta\epsilon^k[x]$ is updated, starting from an all-zero vector. On the other hand, we hold $\Delta\epsilon^k[x]$'s constant while updating ϵ_u and A during the meta-learning stage.

6.4 Experimental Settings

6.4.1 Dataset

Target Tasks We show the effectiveness of the proposed meta-learning method for low resource NMT with extremely limited training examples on five diverse target languages: Romanian (Ro) from WMT'16,[2] Latvian (Lv), Finnish (Fi), Turkish (Tr) from WMT'17,[3] and Korean (Ko) from Korean Parallel Dataset.[4] We use the officially provided train, dev and test splits for all these languages. The statistics of these languages are presented in Table 6.1. We simulate the low-resource translation scenarios by randomly sub-sampling the training set with different sizes.

Source Tasks We use the following languages from Europarl[5]: Bulgarian (Bg), Czech (Cs), Danish (Da), German (De), Greek (El), Spanish (Es), Estonian (Et), French (Fr), Hungarian (Hu), Italian (It), Lithuanian (Lt), Dutch (Nl), Polish (Pl), Portuguese (Pt), Slovak (Sk), Slovene (Sl) and Swedish (Sv), in addition to Russian (Ru)[6] to learn the intilization

[2] http://www.statmt.org/wmt16/translation-task.html
[3] http://www.statmt.org/wmt17/translation-task.html
[4] https://sites.google.com/site/koreanparalleldata/
[5] http://www.statmt.org/europarl/
[6] A subsample of approximately 2M pairs from WMT'17.

Meta-Train	Ro-En		Lv-En		Fi-En		Tr-En		Ko-En	
	zero	finetune	zero	finetune	zero	finetune	zero	finetune	zero	finetune
–		$00.00 \pm .00$		$0.00 \pm .00$		$0.00 \pm .00$		$0.00 \pm .00$		$0.00 \pm .00$
Es	9.20	$15.71 \pm .22$	2.23	$4.65 \pm .12$	2.73	$5.55 \pm .08$	1.56	$4.14 \pm .03$	0.63	$1.40 \pm .09$
Es Fr	12.35	$17.46 \pm .41$	2.86	$5.05 \pm .04$	3.71	$6.08 \pm .01$	2.17	$4.56 \pm .20$	0.61	$1.70 \pm .14$
Es Fr It Pt	13.88	$18.54 \pm .19$	3.88	$5.63 \pm .11$	4.93	$6.80 \pm .04$	2.49	$4.82 \pm .10$	0.82	$1.90 \pm .07$
De Ru	10.60	$16.05 \pm .31$	5.15	$7.19 \pm .17$	6.62	$7.98 \pm .22$	3.20	$6.02 \pm .11$	1.19	$2.16 \pm .09$
Es Fr It Pt De Ru	15.93	$20.00 \pm .27$	6.33	$7.88 \pm .14$	7.89	$9.14 \pm .05$	3.72	$6.02 \pm .13$	1.28	$2.44 \pm .11$
All	18.12	$\mathbf{22.04 \pm .23}$	9.58	$\mathbf{10.44 \pm .17}$	11.39	$\mathbf{12.63 \pm .22}$	5.34	$\mathbf{8.97 \pm .08}$	1.96	$\mathbf{3.97 \pm .10}$
Full Supervised	31.76		15.15		20.20		13.74		5.97	

Table 6.2: BLEU Scores w.r.t. the source task set for all five target tasks.

for fine-tuning. In our experiments, different combinations of source tasks are explored to see the effects from the source tasks.

Validation We pick either Ro-En or Lv-En as a validation set for meta-learning and test the generalization capability on the remaining target tasks. This allows us to study the strict form of meta-learning, in which target tasks are unknown during both training and model selection.

Preprocessing and ULR Initialization As described in §6.3.3, we initialize the query embedding vectors $\epsilon^k_{\text{query}}$ of all the languages. For each language, we use the monolingual corpora built from Wikipedia[7] and the parallel corpus. The concatenated corpus is first tokenized and segmented using byte-pair encoding (BPE, Sennrich et al., 2016), resulting in $40,000$ subwords for each language. We then estimate word vectors using fastText (Bojanowski et al., 2017) and align them across all the languages in an unsupervised way using MUSE (**?**) to get multilingual word vectors. We use the multilingual word vectors of the 20,000 most frequent words in English to form the universal embedding matrix ϵ_u.

6.4.2 Model and Learning

Model We utilize the recently proposed Transformer (Vaswani et al., 2017) as an underlying NMT system. We implement Transformer in this paper based on (Gu et al., 2018a)[8] and modify it to use the universal lexical representation from §6.3.3. We use the default

[7] We use the most recent Wikipedia dump (2018.5) from https://dumps.wikimedia.org/backup-index.html.
[8] https://github.com/salesforce/nonauto-nmt

set of hyperparameters ($d_{\text{model}} = d_{\text{hidden}} = 512$, $n_{\text{layer}} = 6$, $n_{\text{head}} = 8$, $n_{\text{batch}} = 4000$, $t_{\text{warmup}} = 16000$) for all the language pairs and across all the experimental settings. We refer the readers to (Vaswani et al., 2017; Gu et al., 2018a) for the details of the model. However, since the proposed meta-learning method is model-agnostic, it can be easily extended to any other NMT architectures, e.g. RNN-based sequence-to-sequence models with attention (Bahdanau et al., 2014).

Learning We meta-learn using various sets of source languages to investigate the effect of source task choice. For each episode, by default, we use a single gradient step of language-specific learning with Adam (Kingma and Ba, 2014) per computing the meta-gradient, which is computed by the first-order approximation in Eq. (6.3).

For each target task, we sample training examples to form a low-resource task. We build tasks of 4k, 16k, 40k and 160k English tokens for each language. We randomly sample the training set five times for each experiment and report the average score and its standard deviation. Each fine-tuning is done on a training set, early-stopped on a validation set and evaluated on a test set. In default without notation, datasets of 16k tokens are used.

Fine-tuning Strategies The transformer consists of three modules; embedding, encoder and decoder. We update all three modules during meta-learning, but during fine-tuning, we can selectively tune only a subset of these modules. Following (Zoph et al., 2016), we consider three fine-tuning strategies; (1) fine-tuning all the modules (all), (2) fine-tuning the embedding and encoder, but freezing the parameters of the decoder (emb+enc) and (3) fine-tuning the embedding only (emb).

6.5 Results

vs. Multilingual Transfer Learning We meta-learn the initial models on all the source tasks using either Ro-En or Lv-En as a validation task. We also train the initial models to be multilingual translation systems. We fine-tune them using the four target tasks (Ro-En, Lv-En, Fi-En and Tr-En; 16k tokens each) and compare the proposed meta-learning strategy

Source (Tr)	google mülteciler iccin 11 milyon dolar toplamak üzere bağıcs ecslecstirme kampanyasını bacslattı .
Target	google launches donation-matching campaign to raise $ 11 million for refugees .
Meta-0	google refugee fund for usd 11 million has launched a campaign for donation .
Meta-16k	google has launched a campaign to collect $ 11 million for refugees .
Source (Ko)	이번에 체포되어 기소된 사람들 중에는 퇴역한 군 고위관리 , 언론인 , 정치인 , 경제인 등이 포함됐다
Target	among the suspects are retired military officials , journalists , politicians , businessmen and others .
Meta-0	last year , convicted people , among other people , of a high-ranking army of journalists in economic and economic policies , were included .
Meta-16k	the arrested persons were included in the charge , including the military officials , journalists , politicians and economists .

Table 6.3: Sample translations for Tr-En and Ko-En highlight the impact of fine-tuning which results in syntactically better formed translations. We highlight tokens of interest in terms of reordering.

and the multilingual, transfer learning strategy. As presented in Fig. 6.3, the proposed learning approach significantly outperforms the multilingual, transfer learning strategy across all the target tasks regardless of which target task was used for early stopping. We also notice that the emb+enc strategy is most effective for both meta-learning and transfer learning approaches. With the proposed meta-learning and emb+enc fine-tuning, the final NMT systems trained using only a fraction of all available training examples achieve 2/3 (Ro-En) and 1/2 (Lv-En, Fi-En and Tr-En) of the BLEU score achieved by the models trained with full training sets.

Impact of Validation Tasks Similarly to training any other neural network, meta-learning still requires early-stopping to avoid overfitting to a specific set of source tasks. In doing so, we observe that the choice of a validation task has non-negligible impact on the final performance. For instance, as shown in Fig. 6.3, Fi-En benefits more when Ro-En is used for validation, while the opposite happens with Tr-En. The relationship between the task similarity and the impact of a validation task must be investigated further in the future.

Training Set Size We vary the size of the target task's training set and compare the proposed meta-learning strategy and multilingual, transfer learning strategy. We use the emb+enc fine-tuning on Ro-En and Fi-En. Fig. 6.4 demonstrates that the meta-learning approach is more robust to the drop in the size of the target task's training set. The gap between the meta-learning and transfer learning grows as the size shrinks, confirming the effectiveness of the proposed approach on extremely low-resource language pairs.

Impact of Source Tasks In Table 6.2, we present the results on all five target tasks obtained while varying the source task set. We first see that it is always beneficial to use more source tasks. Although the impact of adding more source tasks varies from one language to another, there is up to $2\times$ improvement going from one source task to 18 source tasks (Lv-En, Fi-En, Tr-En and Ko-En). The same trend can be observed even without any fine-tuning (i.e., unsupervised translation, (Lample et al., 2018; Artetxe et al., 2018)). In addition, the choice of source languages has different implications for different target languages. For instance, Ro-En benefits more from $\{$Es, Fr, It, Pt$\}$ than from $\{$De, Ru$\}$, while the opposite effect is observed with all the other target tasks.

Training Curves The benefit of meta-learning over multilingual translation is clearly demonstrated when we look at the training curves in Fig. 6.5. With the multilingual, transfer learning approach, we observe that training rapidly saturates and eventually degrades, as the model overfits to the source tasks. MetaNMT on the other hand continues to improve and never degrades, as the meta-objective ensures that the model is adequate for fine-tuning on target tasks rather than for solving the source tasks.

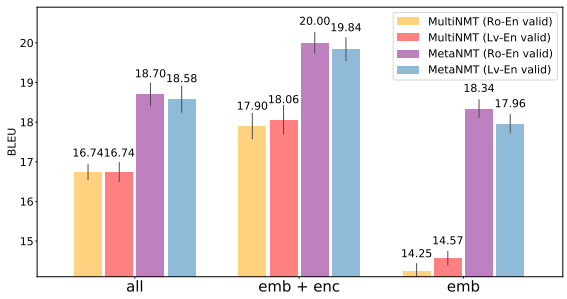
Sample Translations We present some sample translations from the tested models in Table 6.3. Inspecting these examples provides the insight into the proposed meta-learning algorithm. For instance, we observe that the meta-learned model without any fine-tuning produces a word-by-word translation in the first example (Tr-En), which is due to the successful use of the universal lexcial representation and the meta-learned initialization. The system however cannot reorder tokens from Turkish to English, as it has not seen any training example of Tr-En. After seeing around 600 sentence pairs (16K English tokens), the

model rapidly learns to correctly reorder tokens to form a better translation. A similar phenomenon is observed in the Ko-En example. These cases could be found across different language pairs.

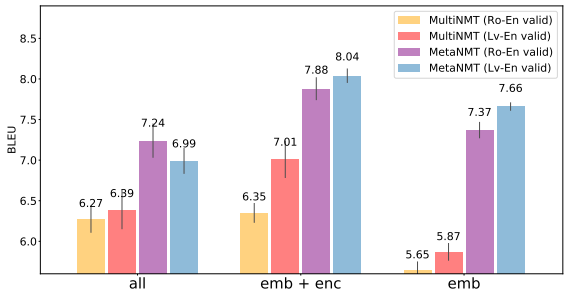
6.6 Conclusion

In this paper, we proposed a meta-learning algorithm for low-resource neural machine translation that exploits the availability of high-resource languages pairs. We based the proposed algorithm on the recently proposed model-agnostic meta-learning and adapted it to work with multiple languages that do not share a common vocabulary using the technique of universal lexcal representation, resulting in MetaNMT. Our extensive evaluation, using 18 high-resource source tasks and 5 low-resource target tasks, has shown that the proposed MetaNMT significantly outperforms the existing approach of multilingual, transfer learning in low-resource neural machine translation across all the language pairs considered.

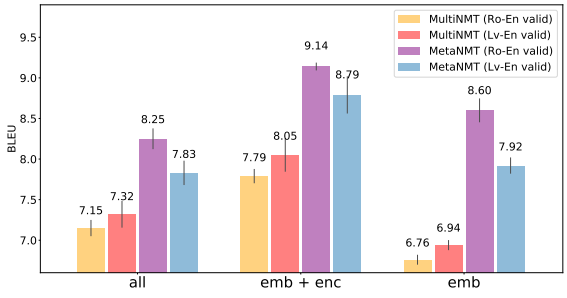
The proposed approach opens new opportunities for neural machine translation. First, it is a principled framework for incorporating various extra sources of data, such as source- and target-side monolingual corpora. Second, it is a generic framework that can easily accommodate existing and future neural machine translation systems.



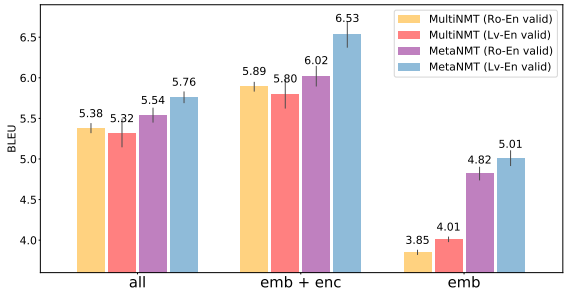
(a) Ro-En



(b) Lv-En



(c) Fi-En



(d) Tr-En

Figure 6.3: BLEU scores reported on test sets for {Ro, Lv, Fi, Tr} to En, where each model is first learned from 6 source tasks (Es, Fr, It, Pt, De, Ru) and then fine-tuned on randomly sampled training sets with around 16,000 English tokens per run. The error bars show the standard deviation calculated from 5 runs.

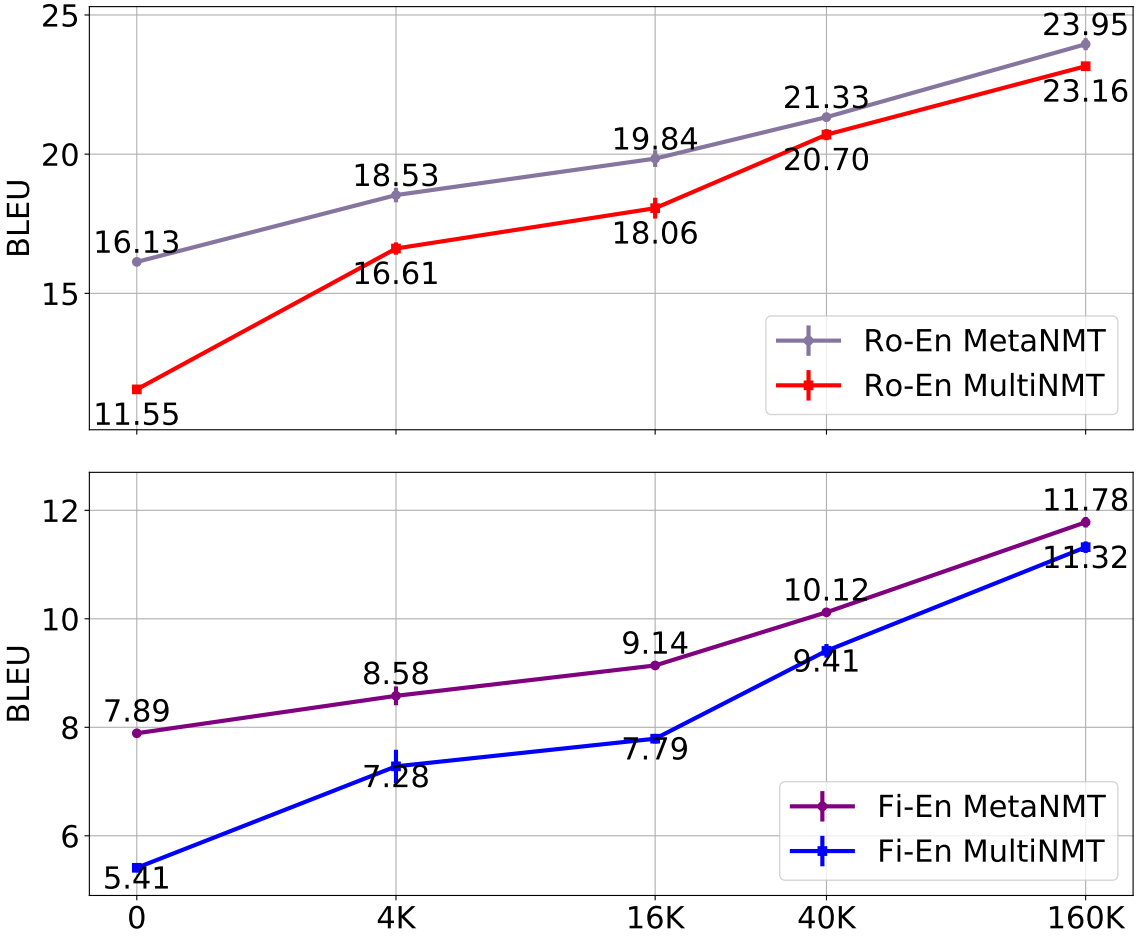


Figure 6.4: BLEU Scores w.r.t. the size of the target task's training set.

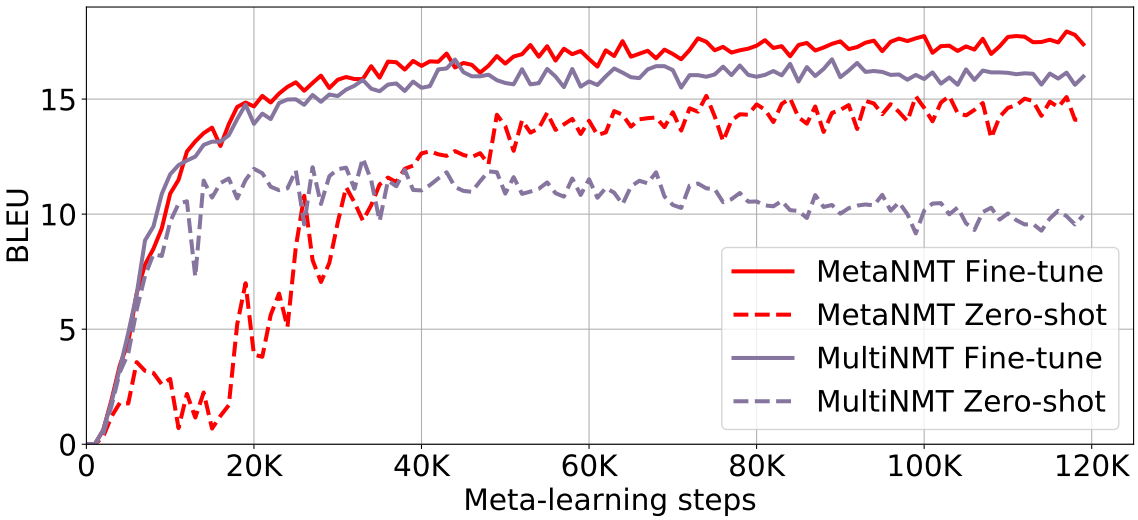


Figure 6.5: The learning curves of BLEU scores on the validation task (Ro-En).

Part II

Decoding-Efficient Neural Machine Translation

Chapter 7

Trainable Decoding for Neural Machine Translation

7.1 Introduction

Neural machine translation has recently become a method of choice in machine translation research. Besides its success in traditional settings of machine translation, that is one-to-one translation between two languages, (Sennrich et al., 2016; Chung et al., 2016), neural machine translation has ventured into more sophisticated settings of machine translation. For instance, neural machine translation has successfully proven itself to be capable of handling subword-level representation of sentences (Lee et al., 2016; Luong and Manning, 2016; Sennrich et al., 2015b; Costa-Jussa and Fonollosa, 2016; Ling et al., 2015). Furthermore, several research groups have shown its potential in seamlessly handling multiple languages (Dong et al., 2015; Luong et al., 2015a; Firat et al., 2016a,b; Lee et al., 2016; Ha et al., 2016b; Viégas et al., 2016).

A typical scenario of neural machine translation starts with training a model to maximize its log-likelihood. That is, we often train a model to maximize the conditional probability of a reference translation given a source sentence over a large parallel corpus. Once the model is trained in this way, it defines the conditional distribution over all possible translations given a source sentence, and the task of translation becomes equivalent to finding a translation to which the model assigns the highest conditional probability. Since it

is computationally intractable to do so exactly, it is a usual practice to resort to approximate search/decoding algorithms such as greedy decoding or beam search. In this scenario, we have identified two points where improvements could be made. They are (1) training (including the selection of a model architecture) and (2) decoding.

Much of the research on neural machine translation has focused solely on the former, that is, on improving the model architecture. Neural machine translation started with with a simple encoder-decoder architecture in which a source sentence is encoded into a single, fixed-size vector (Cho et al., 2014b; Sutskever et al., 2014; Kalchbrenner and Blunsom, 2013). It soon evolved with the attention mechanism (Bahdanau et al., 2014). A few variants of the attention mechanism, or its regularization, have been proposed recently to improve both the translation quality as well as the computational efficiency (Luong et al., 2015b; Cohn et al., 2016; Tu et al., 2016b). More recently, convolutional networks have been adopted either as a replacement of or a complement to a recurrent network in order to efficiently utilize parallel computing (Kalchbrenner et al., 2016; Lee et al., 2016; Gehring et al., 2016).

On the aspect of decoding, only a few research groups have tackled this problem by incorporating a target decoding algorithm into training. Wiseman and Rush (2016) and Shen et al. (2015) proposed a learning algorithm tailored for beam search. Ranzato et al. (2015) and (Bahdanau et al., 2016) suggested to use a reinforcement learning algorithm by viewing a neural machine translation model as a policy function. Investigation on decoding alone has, however, been limited. Cho (2016) showed the limitation of greedy decoding by simply injecting unstructured noise into the hidden state of the neural machine translation system. Tu et al. (2016a) similarly showed that the exactness of beam search does not correlate well with actual translation quality, and proposed to augment the learning cost function with reconstruction to alleviate this problem. Li et al. (2016a) proposed a modification to the existing beam search algorithm to improve its exploration of the translation space.

In this paper, we tackle the problem of decoding in neural machine translation by introducing a concept of *trainable greedy decoding*. Instead of manually designing a new decoding algorithm suitable for neural machine translation, we propose to learn a decoding

algorithm with an arbitrary decoding objective. More specifically, we introduce a neural-network-based decoding algorithm that works on an already-trained neural machine translation system by observing and manipulating its hidden state. We treat such a neural network as an agent with a deterministic, continuous action and train it with a variant of the deterministic policy gradient algorithm (Silver et al., 2014).

We extensively evaluate the proposed trainable greedy decoding on four language pairs (En-Cs, En-De, En-Ru and En-Fi; in both directions) with two different decoding objectives; sentence-level BLEU and negative perplexity. By training such trainable greedy decoding using deterministic policy gradient with the proposed critic-aware actor learning, we observe that we can improve decoding performance with minimal computational overhead. Furthermore, the trained actors are found to improve beam search as well, suggesting a future research direction in extending the proposed idea of trainable decoding for more sophisticated underlying decoding algorithms.

7.2 Background

7.2.1 Neural Machine Translation

Neural machine translation is a special case of conditional recurrent language modeling, where the source and target are natural language sentences. Let us use $X = \{x_1, \dots, x_{T_s}\}$ and $Y = \{y_1, \dots, y_T\}$ to denote source and target sentences, respectively. Neural machine translation then models the target sentence given the source sentence as: $p(Y|X) = \prod_{t=1}^{T} p(y_t|y_{<t}, X)$. Each term on the r.h.s. of the equation above is modelled as a composite of two parametric functions:

$$p(y_t|y_{<t}, X) \propto \exp\left(g\left(y_t, z_t; \theta_g\right)\right),$$

where $z_t = f(z_{t-1}, y_{t-1}, e_t(X; \theta_e); \theta_f)$. g is a read-out function that transforms the hidden state z_t into the distribution over all possible symbols, and f is a recurrent function that compresses all the previous target words $y_{<t}$ and the time-dependent representation

$e_t(X;\theta_e)$ of the source sentence X. This time-dependent representation e_t is often implemented as a recurrent network encoder of the source sentence coupled with an attention mechanism (Bahdanau et al., 2014).

Maximum Likelihood Learning We train a neural machine translation model, or equivalently estimate the parameters θ_g, θ_f and θ_e, by maximizing the log-probability of a reference translation $\hat{Y} = \{\hat{y}_1, ..., \hat{y}_T\}$ given a source sentence. That is, we maximize the log-likelihood function:

$$J^{\text{ML}}(\theta_g, \theta_f, \theta_e) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\log p_\theta(\hat{y}_t^n|\hat{y}_{<t}^n, X^n),$$

given a training set consisting of N source-target sentence pairs. It is important to note that this maximum likelihood learning does not take into account how a trained model would be used. Rather, it is only concerned with learning a distribution over all possible translations.

7.2.2 Decoding

Once the model is trained, either by maximum likelihood learning or by any other recently proposed algorithms (Wiseman and Rush, 2016; Shen et al., 2015; Bahdanau et al., 2016; Ranzato et al., 2015), we can let the model translate a given sentence by finding a translation that maximizes

$$\hat{Y} = \operatorname*{argmax}_Y \log p_\theta(Y|X),$$

where $\theta = (\theta_g, \theta_f, \theta_e)$. This is, however, computationally intractable, and it is a usual practice to resort to approximate decoding algorithms.

Greedy Decoding One such approximate decoding algorithm is greedy decoding. In greedy decoding, we follow the conditional dependency path and pick the symbol with the highest conditional probability so far at each node. This is equivalent to picking the best symbol one at a time from left to right in conditional language modelling. A decoded

translation of greedy decoding is $\hat{Y} = (\hat{y}_1, \ldots, \hat{y}_T)$, where

$$\hat{y}_t = \underset{y \in V}{\operatorname{argmax}} \log p_\theta(y | \hat{y}_{<t}, X). \tag{7.1}$$

Despite its preferable computational complexity $O(|V| \times T)$, greedy decoding has been over time found to be undesirably sub-optimal.

Beam Search Beam search keeps $K > 1$ hypotheses, unlike greedy decoding which keeps only a single hypothesis during decoding. At each time step t, beam search picks K hypotheses with the highest scores ($\prod_{t'=1}^{t} p(y_t | y_{<t}, X)$). When all the hypotheses terminate (outputting the end-of-the-sentence symbol), it returns the hypothesis with the highest log-probability. Despite its superior performance compared to greedy decoding, the computational complexity grows linearly w.r.t. the size of beam K, which makes it less preferable especially in the production environment.

7.3 Trainable Greedy Decoding

7.3.1 Many Decoding Objectives

Although we have described decoding in neural machine translation as a maximum-a-posteriori estimation in $\log p(Y|X)$, this is not necessarily the only nor the desirable decoding objective.

First, each potential scenario in which neural machine translation is used calls for a unique decoding objective. In simultaneous translation/interpretation, which has recently been studied in the context of neural machine translation (Gu et al., 2016b), the decoding objective is formulated as a trade-off between the translation quality and delay. On the other hand, when a machine translation system is used as a part of a larger information extraction system, it is more important to correctly translate named entities and events than to translate syntactic function words. The decoding objective in this case must account for how the translation is used in subsequent modules in a larger system.

Second, the conditional probability assigned by a trained neural machine translation

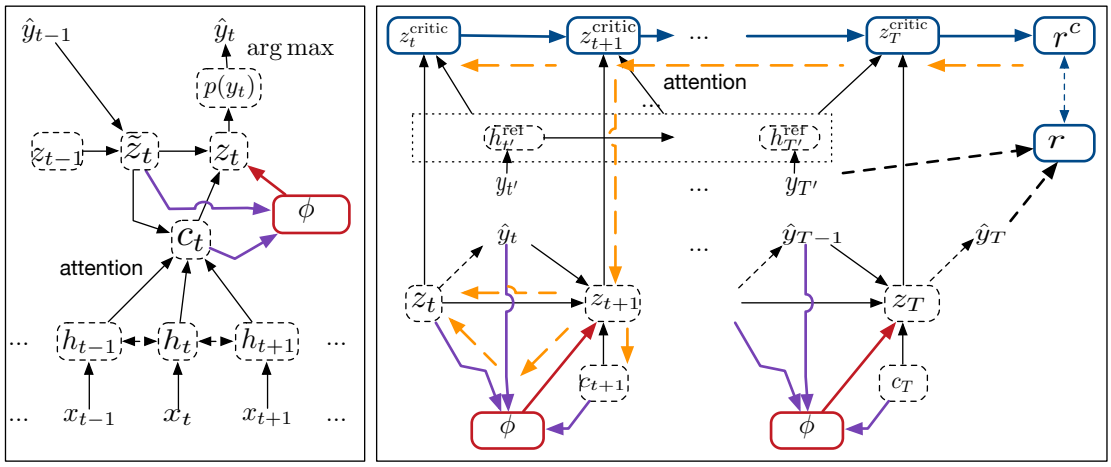


Figure 7.1: Graphical illustrations of the trainable greedy decoding. The left panel shows a single step of the actor interacting with the underlying neural translation model, and The right panel the interaction among the underlying neural translation system (dashed-border boxes), actor (red-border boxes), and critic (blue-border boxes). The solid arrows indicate the forward pass, and the dashed yellow arrows the actor's backward pass. The dotted-border box shows the use of a reference translation.

model does not necessarily reflect our perception of translation quality. Although Cho (2016) provided empirical evidence of high correlation between the log-probability and BLEU, a *de facto* standard metric in machine translation, there have also been reports on large mismatch between the log-probability and BLEU. For instance, Tu et al. (2016a) showed that beam search with a very large beam, which is supposed to find translations with better log-probabilities, suffers from pathological translations of very short length, resulting in low translation quality. This calls for a way to design or *learn* a decoding algorithm with an objective that is more directly correlated to translation quality.

In short, there is a significant need for designing multiple decoding algorithms for neural machine translation, regardless of how it was trained. It is however non-trivial to manually design a new decoding algorithm with an arbitrary objective. This is especially true with neural machine translation, as the underlying structure of the decoding/search process – the high-dimensional hidden state of a recurrent network – is accessible but not interpretable. Instead, in the remainder of this section, we propose our approach of *trainable greedy decoding*.

7.3.2 Trainable Greedy Decoding

We start from the noisy, parallel approximate decoding (NPAD) algorithm proposed in (Cho, 2016). The main idea behind NPAD algorithm is that a better translation with a higher log-probability may be found by injecting unstructured noise in the transition function of a recurrent network. That is,

$$z_t = f(z_{t-1} + \epsilon_t, y_{t-1}, e_t(X; \theta_e); \theta_f),$$

where $\epsilon_t \sim \mathcal{N}(0, (\sigma_0/t)^2)$. NPAD avoids potential degradation of translation quality by running such a noisy greedy decoding process multiple times in parallel. An important lesson of NPAD algorithm is that there exists a decoding strategy with the asymptotically same computational complexity that results in a better translation quality, and that such a better translation can be found by manipulating the hidden state of the recurrent network.

In this work, we propose to significantly extend NPAD by replacing the unstructured noise ϵ_t with a parametric function approximator, or an agent, π_ϕ. This agent takes as input the previous hidden state z_{t-1}, previously decoded word $\hat{y}_{t-1}$ and the time-dependent context vector $e_t(X; \theta_e)$ and outputs a real-valued vectorial action $a_t \in \mathbb{R}^{\dim(z_t)}$. Such an agent is trained such that greedy decoding with the agent finds a translation that maximizes any predefined, arbitrary decoding objective, while the underlying neural machine translation model is pretrained and fixed. Once the agent is trained, we generate a translation given a source sentence by greedy decoding however augmented with this agent. We call this decoding strategy *trainable greedy decoding*.

Related Work: Soothsayer prediction function Independently from and concurrently with our work here, Li et al. (2017) proposed, just two weeks earlier, to train a neural network that predicts an arbitrary decoding objective given a source sentence and a partial hypothesis, or a prefix of translation, and to use it as an auxiliary score in beam search. For training such a network, referred to as a Q network in their paper, they generate each training example by either running beam search or using a ground-truth translation (when appropriate) for each source sentence. This approach allows one to use an arbitrary decoding objective, but it still relies heavily on the log-probability of the underlying neural

translation system in actual decoding. We expect a combination of these and our approaches may further improve decoding for neural machine translation in the future.

7.3.3 Learning and Challenges

While all the parameters—θ_g, θ_f and θ_e— of the underlying neural translation model are fixed, we only update the parameters ϕ of the agent π. This ensures the generality of the pre-trained translation model, and allows us to train multiple trainable greedy decoding agents with different decoding objectives, maximizing the utility of a single trained translation model.

Let us denote by R our arbitrary decoding objective as a function that scores a translation generated from trainable greedy decoding. Then, our learning objective for trainable greedy decoding is

$$J^{\mathrm{A}}(\phi) = \mathbb{E}^{\hat{Y}=G_\pi(X)}_{X\sim D}\left[R(\hat{Y})\right],$$

where we used $G_\pi(X)$ as a shorthand for trainable greedy decoding with an agent π.

There are two major challenges in learning an agent with such an objective. First, the decoding objective R may not be differentiable with respect to the agent. Especially because our goal is to accommodate an arbitrary decoding objective, this becomes a problem. For instance, BLEU, a standard quality metric in machine translation, is a piece-wise linear function with zero derivatives almost everywhere. Second, the agent here is a real-valued, deterministic policy with a very high-dimensional action space (1000s of dimensions), which is well known to be difficult. In order to alleviate these difficulties, we propose to use a variant of the deterministic policy gradient algorithm (Silver et al., 2014; Lillicrap et al., 2015).

7.4 Deterministic Policy Gradient with Critic-Aware Actor Learning

7.4.1 Deterministic Policy Gradient for Trainable Greedy Decoding

It is highly unlikely for us to have access to the gradient of an arbitrary decoding objective R with respect to the agent π, or its parameters ϕ. Furthermore, we cannot estimate it stochastically because our policy π is defined to be deterministic without a predefined nor learned distribution over the action. Instead, following (Silver et al., 2014; Lillicrap et al., 2015), we use a parametric, differentiable approximator, called a critic R^c, for the non-differentiable objective R. We train the critic by minimizing

$$J^{\mathrm{C}}(\psi) = \mathbb{E}_{X\sim D}^{\hat{Y}=G_\pi(X)}\left[R^c_\psi(z_{1:T}) - R(\hat{Y})\right]^2.$$

The critic observes the state-action sequence of the agent π via the modified hidden states $(z_1, \ldots, z_T)$ of the recurrent network, and predicts the associated decoding objective. By minimizing the mean squared error above, we effectively encourage the critic to approximate the non-differentiable objective as closely as possible in the vicinity of the state-action sequence visited by the agent.

We implement the critic R^c as a recurrent network, similarly to the underlying neural machine translation system. This implies that we can compute the derivative of the predicted decoding objective with respect to the input, that is, the state-action sequence $z_{1:T}$, which allows us to update the actor π, or equivalently its parameters ϕ, to maximize the predicted decoding objective. Effectively we avoid the issue of non-differentiability of the original decoding objective by working with its proxy.

With the critic, the learning objective of the actor is now to maximize not the original decoding objective R but its proxy R^{C} such that

$$\hat{J}^{\mathrm{A}}(\phi) = \mathbb{E}_{X\sim D}^{\hat{Y}=G_\pi(X)}\left[R^{\mathrm{C}}(\hat{Y})\right].$$

Unlike the original objective, this objective function is fully differentiable with respect to

the agent π. We thus use a usual stochastic gradient descent algorithm to train the agent, while simultaneously training the critic. We do so by alternating between training the actor and critic. Note that we maximize the return of a full episode rather than the Q value, unlike usual approaches in reinforcement learning.

7.4.2 Critic-Aware Actor Learning

Challenges The most apparent challenge for training such a deterministic actor with a large action space is that most of action configurations will lead to zero return. It is also not trivial to devise an efficient exploration strategy with a deterministic actor with real-valued actions. This issue has however turned out to be less of a problem than in a usual reinforcement learning setting, as the state and action spaces are well structured thanks to pretraining by maximum likelihood learning. As observed by Cho (2016), any reasonable perturbation to the hidden state of the recurrent network generates a reasonable translation which would receive again a reasonable return.

Although this property of dense reward makes the problem of trainable greedy decoding more manageable, we have observed other issues during our preliminary experiment with the vanilla deterministic policy gradient. In order to avoid these issues that caused instability, we propose the following modifications to the vanilla algorithm.

Critic-Aware Actor Learning A major goal of the critic is not to estimate the return of a given episode, but to estimate the gradient of the return evaluated given an episode. In order to do so, the critic must be trained, or presented, with state-action sequences $z_{1:T'}$ similar though not identical to the state-action sequence generated by the current actor π. This is achieved, in our case, by injecting unstructured noise to the action at each time step, similar to (Heess et al., 2015):

$$\tilde{a}_t = \phi(z_t, a_{t-1}) + \sigma \cdot \epsilon, \tag{7.2}$$

where ϵ is a zero-mean, unit-variance normal variable. This noise injection procedure is mainly used when training the critic.

We have however observed that the quality of the reward and its gradient estimate of

the critic is very noisy even when the critic was trained with this kind of noisy actor. This imperfection of the critic often led to the instability in training the actor in our preliminary experiments. In order to avoid this, we describe here a technique which we refer to as *critic-aware actor gradient estimation*.

Instead of using the point estimate $\frac{\partial R^c}{\partial \phi}$ of the gradient of the predicted objective with respect to the actor's parameters ϕ, we propose to use the expected gradient of the predicted objective with respect to the critic-aware distribution Q. That is,

$$\mathbb{E}_Q \left[\frac{\partial R^c_\psi}{\partial \phi} \right], \tag{7.3}$$

where we define the critic-aware distribution Q as

$$Q(\epsilon) \propto \exp(\underbrace{-(R^c_\psi - R)^2/\tau)}_{\text{Critic-awareness}} \exp(\underbrace{-\frac{\epsilon^2}{2\sigma^2}}_{\text{Locality}}). \tag{7.4}$$

This expectation allows us to incorporate the noisy, non-uniform nature of the critic's approximation of the objective by up-weighting the gradient computed at a point with a higher critic quality and down-weighting the gradient computed at a point with a lower critic quality. The first term in Q reflects this, while the second term ensures that our estimation is based on a small region around the state-action sequence generated by the current, noise-free actor π.

Since it is intractable to compute Eq. (7.3) exactly, we resort to importance sampling with the proposed distribution equal to the second term in Eq. (7.4). Then, our gradient estimate for the actor becomes the sum of the gradients from multiple realizations of the noisy actor in Eq. (7.2), where each gradient is weighted by the quality of the critic $\exp(-(R^c_\phi - R)^2/\tau)$. τ is a hyperparameter that controls the smoothness of the weights. We observed in our preliminary experiment that the use of this critic-aware actor learning significantly stabilizes general learning of both the actor and critic.

Reference Translations for Training the Critic In our setting of neural machine translation, we have access to a reference translation for each source sentence X, unlike in a usual setting of reinforcement learning. By force-feeding the reference translation into the

underlying neural machine translation system (rather than feeding the decoded symbols), we can generate the reference state-action sequence. This sequence is much less correlated with those sequences generated by the actor, and facilitates computing a better estimate of the gradient w.r.t. the critic.

In Alg. 5, we present the complete algorithm. To make the description less cluttered, we only show the version of minibatch size = 1 which can be naturally extended. We also illustrate the proposed trainable greedy decoding and the proposed learning strategy in Fig. 7.1.

7.5 Experimental Settings

We empirically evaluate the proposed trainable greedy decoding on four language pairs – En-De, En-Ru, En-Cs and En-Fi – using a standard attention-based neural machine translation system (Bahdanau et al., 2014). We train underlying neural translation systems using the parallel corpora made available from WMT'15.[1] The same set of corpora are used for trainable greedy decoding as well. All the corpora are tokenized and segmented into subword symbols using byte-pair encoding (BPE) (Sennrich et al., 2015b). We use sentences of length up to 50 subword symbols for MLE training and 200 symbols for trainable decoding. For validation and testing, we use newstest-2013 and newstest-2015, respectively.

7.5.1 Model Architectures and Learning

Underlying NMT Model For each language pair, we implement an attention-based neural machine translation model whose encoder and decoder recurrent networks have 1,028 gated recurrent units (GRU, Cho et al., 2014b) each. Source and target symbols are projected into 512-dimensional embedding vectors. We trained each model for approximately 1.5 weeks using Adadelta (Zeiler, 2012).

Actor π We use a feedforward network with a single hidden layer as the actor. The input is a 2,056-dimensional vector which is the concatenation of the decoder hidden state and

[1] http://www.statmt.org/wmt15/

the time-dependent context vector from the attention mechanism, and it outputs a 1,028-dimensional action vector for the decoder. We use 32 units for the hidden layer with $\tanh$ activations.

Critic R^c The critic is implemented as a variant of an attention-based neural machine translation model that takes a reference translation as a source sentence and a state-action sequence from the actor as a target sentence. Both the size of GRU units and embedding vectors are the same with the underlying model. Unlike a usual neural machine translation system, the critic does not language-model the target sentence but simply outputs a scalar value to predict the true return. When we predict a bounded return, such as sentence BLEU, we use a sigmoid activation at the output. For other unbounded return like perplexity, we use a linear activation.

Learning We train the actor and critic simultaneously by alternating between updating the actor and critic. As the quality of the critic's approximation of the decoding objective has direct influence on the actor's learning, we make ten updates to the critic before each time we update the actor once. We use RMSProp (Tieleman and Hinton, 2012) with the initial learning rates of 2×10^{-6} and 2×10^{-4}, respectively, for the actor and critic.

We monitor the progress of learning by measuring the decoding objective on the validation set. After training, we pick the actor that results in the best decoding objective on the validation set, and test it on the test set.

Decoding Objectives For each neural machine translation model, pretrained using maximum likelihood criterion, we train two trainable greedy decoding actors. One actor is trained to maximize BLEU (or its smoothed version for sentence-level scoring (Lin and Och, 2004)) as its decoding objective, and the other to minimize perplexity (or equivalently the negative log-probability normalized by the length.)

We have chosen the first two decoding objectives for two purposes. First, we demonstrate that it is possible to build multiple trainable decoders with a single underlying model trained using maximum likelihood learning. Second, the comparison between these two objectives provides a glimpse into the relationship between BLEU (the most widely used

automatic metric for evaluating translation systems) and log-likelihood (the most widely used learning criterion for neural machine translation).

Evaluation We test the trainable greedy decoder with both greedy decoding and beam search. Although our decoder is always trained with greedy decoding, beam search in practice can be used together with the actor of the trainable greedy decoder. Beam search is expected to work better especially when our training of the trainable greedy decoder is unlikely to be optimal. In both cases, we report both the perplexity and BLEU.

7.5.2 Results and Analysis

We present the improvements of BLEU and perplexity (or its negation) in Fig. 7.2 for all the language pair-directions. It is clear from these plots that the best result is achieved when the trainable greedy decoder was trained to maximize the target decoding objective. When the decoder was trained to maximize sentence-level BLEU, we see the improvement in BLEU but often the degradation in the perplexity (see the left plots in Fig. 7.2.) On the other hand, when the actor was trained to minimize the perplexity, we only see the improvement in perplexity (see the right plots in Fig. 7.2.) This confirms our earlier claim that it is necessary and desirable to tune for the target decoding objective regardless of what the underlying translation system was trained for, and strongly supports the proposed idea of trainable decoding.

The improvement from using the proposed trainable greedy decoding is smaller when used together with beam search, as seen in Fig. 7.2 (b). However, we still observe statistically significant improvement in terms of BLEU (marked with red stars.) This suggests a future direction in which we extend the proposed trainable greedy decoding to directly incorporate beam search into its training procedure to further improve the translation quality.

It is worthwhile to note that we achieved all of these improvements with negligible computational overhead. This is due to the fact that our actor is a very small, shallow neural network, and that the more complicated critic is thrown away after training. We suspect the effectiveness of such a small actor is due to the well-structured hidden state space of the underlying neural machine translation model which was trained with a large amount of

parallel corpus. We believe this favourable computational complexity makes the proposed method suitable for production-grade neural machine translation (Wu et al., 2016; Crego et al., 2016).

Importance of Critic-Aware Actor Learning In Fig. 7.3, we show sample learning curves with and without the proposed critic-aware actor learning. Both curves were from the models trained under the same condition. Despite a slower start in the early stage of learning, we see that the critic-aware actor learning has greatly stabilized the learning progress. We emphasize that we would not have been able to train all these 16 actors without the proposed critic-aware actor learning.

Examples In Fig. 7.4, we present three examples from Ru-En. We defined the influence as the KL divergence between the conditional distributions without the trainable greedy decoding and with the trainable greedy decoding, assuming the fixed previous hidden state and target symbol. We colored a target word with magenta, when the influence of the trainable greedy decoding is large (> 0.001). Manual inspection of these examples as well as others has revealed that the trainable greedy decoder focuses on fixing prepositions and removing any unnecessary symbol generation. More in-depth analysis is however left as future work.

7.6 Conclusion

We proposed trainable greedy decoding as a way to learn a decoding algorithm for neural machine translation with an arbitrary decoding objective. The proposed trainable greedy decoder observes and manipulates the hidden state of a trained neural translation system, and is trained by a novel variant of deterministic policy gradient, called critic-aware actor learning. Our extensive experiments on eight language pair-directions and two objectives confirmed its validity and usefulness. The proposed trainable greedy decoding is a generic idea that can be applied to any recurrent language modeling, and we anticipate future research both on the fundamentals of the trainable decoding as well as on the applications to more diverse tasks such as image caption generating and dialogue modeling.

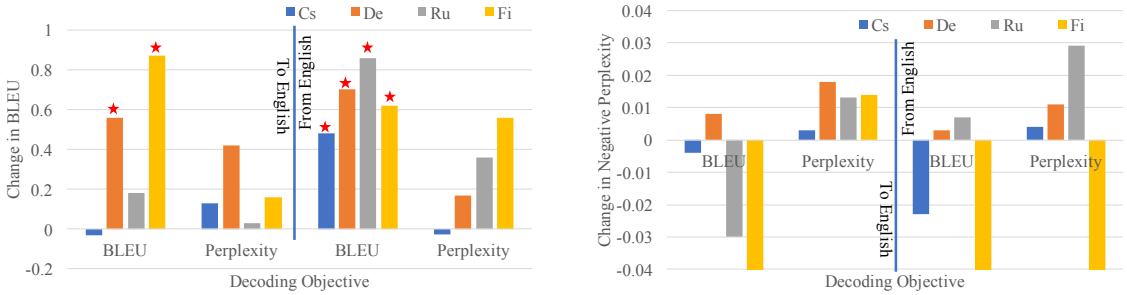
Algorithm 3 Trainable Greedy Decoding

Require: NMT θ, actor ϕ, critic ψ, N_c, N_a, S_c, S_a, τ

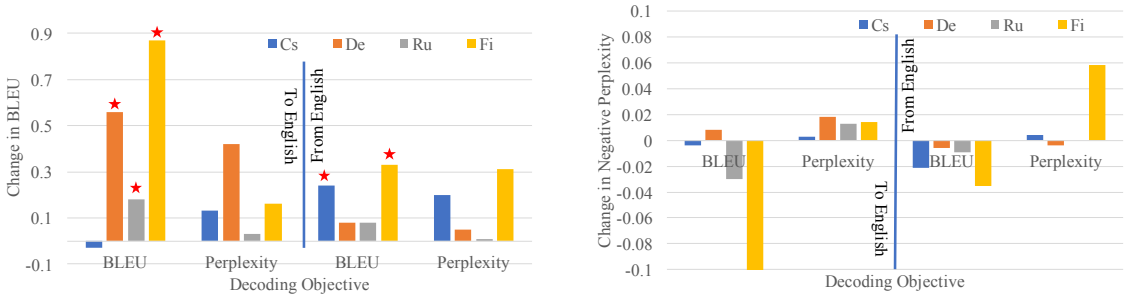
Train θ using MLE on training set D;
Initialize ϕ and ψ;
Shuffle D twice into D_ϕ and D_ψ
while stopping criterion is not met **do**
    for $t = 1 : N_c$ **do**
        Draw a translation pair: $(X, Y) \sim D_\psi$;
        $r, r^c = \text{DECODE}(S_c, X, Y, 1)$
        Update ψ using $\nabla_\psi \sum_k \left(r_k^c - r_k\right)^2 / (S_c + 1)$
    for $t = 1 : N_a$ **do**
        Draw a translation pair: $(X, Y) \sim D_\phi$;
        $r, r^c = \text{DECODE}(S_a, X, Y, 0)$
        Compute $w_k = \exp\left(-\left(r_k^c - r_k\right)^2 / \tau\right)$
        Compute $\tilde{w}_k = w_k / \sum_k w_k$
        Update ϕ using $-\sum_k \left(\tilde{w}_k \cdot \nabla_\phi r_k^c\right)$

Function: $\text{DECODE}(S, X, Y, c)$

$Y_s = \{\}$, $Z_s = \{\}$, $r = \{\}$, $r^c = \{\}$;
for $k = 1 : S$ **do**
    Sample noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ for each action;
    Greedy decoding $\hat{Y}^k = G_{\theta,\phi}(X)$ with ϵ;
    Collect hidden states $z_{1:T}^k$ given X, $\hat{Y}$, θ, ϕ
    $Y_s \leftarrow Y_s \cup \{Y^k\}$
    $Z_s \leftarrow Z_s \cup \{z_{1:T}^k\}$
if $c = 1$ **then**
    Collect hidden states $z_{1:T}$ given X, Y, θ
    $Y_s \leftarrow Y_s \cup \{Y\}$
    $Z_s \leftarrow Z_s \cup \{z_{1:T}\}$
for $\hat{Y}, Z \in Y_s, Z_s$ **do**
    Compute the critic output $r^c \leftarrow R_\psi^c(Z, \hat{Y})$
    Compute true reward $r \leftarrow R(Y, \hat{Y})$
return r, r^c



(a) Trainable Greedy Decoding



(b) Beam Search + Trainable Greedy Decoding

Figure 7.2: The plots draw the improvements by the trainable greedy decoding on the test set. The x-axes correspond to the objectives used to train trainable greedy decoding, and the y-axes to the changes in the achieved objectives (BLEU for the figures on the left, and negative perplexity on the right.) The top row (a) shows the cases when the trainable greedy decoder is used on its own, and the bottom row (b) when it is used together with beam search. When training and evaluation are both done with BLEU, we test the statistical significance (Koehn, 2004), and we mark significant cases with red stars ($p < 0.05$.) The underlying neural machine translation models achieved the BLEU scores of 14.49/16.20 for En-Cs, 18.90/21.20 for Cs-En, 18.97/21.33 for En-De, 21.63/24.46 for De-En, 16.97/19.68 for En-Ru, 21.06/23.34 for Ru-En, 7.53/8.82 for En-Fi and 9.79/11.03 for Fi-En (greedy/beam).

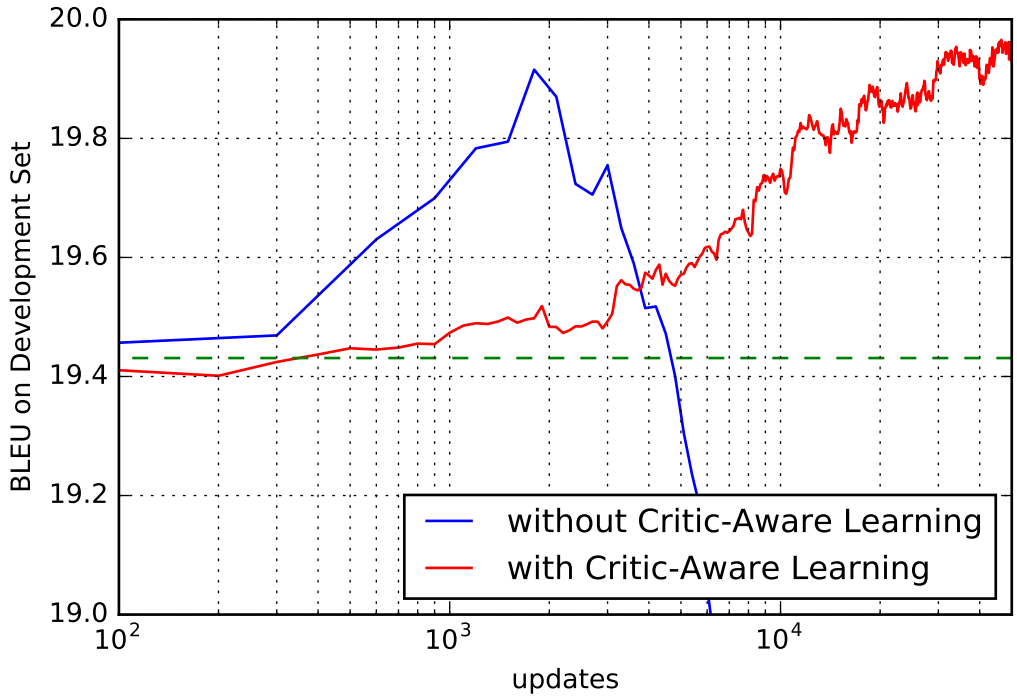


Figure 7.3: Comparison of greedy BLEU scores whether using the critic-aware exploration or not on Ru-En Dataset. The green line means the BLEU score achieved by greedy decoding from the underlying NMT model.

```
(a) S:  Главное зеркало инфракрасного космического телескопа имеет диаметр 6,5 метров
    T:  The primary mirror of the infrared space telescope has a diameter of 6.5 metres .
    G:  The main mirror of the infrared spaceboard has a diameter 6.5 m .
    A:  The main mirror of the infrared space-type telescope has a diameter of 6.5 meters .

(b) S:  Еще один пункт - это дать им понять , что они должны вести себя онлайн так же , как делают это оффлайн .
    T:  Another point is to make them see that they must behave online as they do offline .
    G:  Another option is to give them a chance to behave online as well as do this offline .
    A:  Another option is to give them to know that they must behave online as well as offline .

(c) S:  Возможен ли долговременный мир между арабами и израильтянами на Ближнем Востоке ?
    T:  Can there ever be a lasting peace between Arabs and Jews in the Middle East ?
    G:  Can the Long-term Peace be Out of the Middle East ?
    A:  Can the Long-term Peace be between Arabs and Israelis in the Middle East ?
```

Figure 7.4: Three Ru-En examples in which the difference between the trainable greedy decoding (A) and the conventional greedy decoding (G) is large. Each step is marked with magenta, when the actor significantly influenced the output distribution.

Chapter 8

Non-Autoregressive Neural Machine Translation

8.1 Introduction

Neural network based models outperform traditional statistical models for machine translation (MT) (Bahdanau et al., 2014; Luong et al., 2015b). However, state-of-the-art neural models are much slower than statistical MT approaches at inference time (Wu et al., 2016). Both model families use *autoregressive* decoders that operate one step at a time: they generate each token conditioned on the sequence of tokens previously generated. This process is not parallelizable, and, in the case of neural MT models, it is particularly slow because a computationally intensive neural network is used to generate each token.

While several recently proposed models avoid recurrence at train time by leveraging convolutions (Kalchbrenner et al., 2016; Gehring et al., 2017; Kaiser et al., 2017a) or self-attention (Vaswani et al., 2017) as more-parallelizable alternatives to recurrent neural networks (RNNs), use of autoregressive decoding makes it impossible to take full advantage of parallelism during inference.

We introduce a non-autoregressive translation model based on the Transformer network (Vaswani et al., 2017). We modify the encoder of the original Transformer network by adding a module that predicts *fertilities*, sequences of numbers that form an important component of many traditional machine translation models (Brown et al., 1993). These

fertilities are supervised during training and provide the decoder at inference time with a globally consistent plan on which to condition its simultaneously computed outputs.

8.2 Background

8.2.1 Autoregressive Neural Machine Translation

Given a source sentence $X = \{x_1, ..., x_{T'}\}$, a neural machine translation model factors the distribution over possible output sentences $Y = \{y_1, ..., y_T\}$ into a chain of conditional probabilities with a left-to-right causal structure:

$$p_{\mathcal{AR}}(Y|X;\theta) = \prod_{t=1}^{T+1} p(y_t|y_{0:t-1}, x_{1:T'};\theta), \tag{8.1}$$

where the special tokens y_0 (e.g. $\langle\text{bos}\rangle$) and y_{T+1} (e.g. $\langle\text{eos}\rangle$) are used to represent the beginning and end of all target sentences. These conditional probabilities are parameterized using a neural network. Typically, an encoder-decoder architecture (Sutskever et al., 2014) with a unidirectional RNN-based decoder is used to capture the causal structure of the output distribution.

Maximum Likelihood training Choosing to factorize the machine translation output distribution autoregressively enables straightforward maximum likelihood training with a cross-entropy loss applied at each decoding step:

$$\mathcal{L}_{\text{ML}} = \log p_{\mathcal{AR}}(Y|X;\theta) = \sum_{t=1}^{T+1} \log p(y_t|y_{0:t-1}, x_{1:T'};\theta). \tag{8.2}$$

This loss provides direct supervision for each conditional probability prediction.

Autoregressive NMT without RNNs Since the entire target translation is known at training time, the calculation of later conditional probabilities (and their corresponding losses) does not depend on the output words chosen during earlier decoding steps. Even though decoding must remain entirely sequential during inference, models can take advantage of

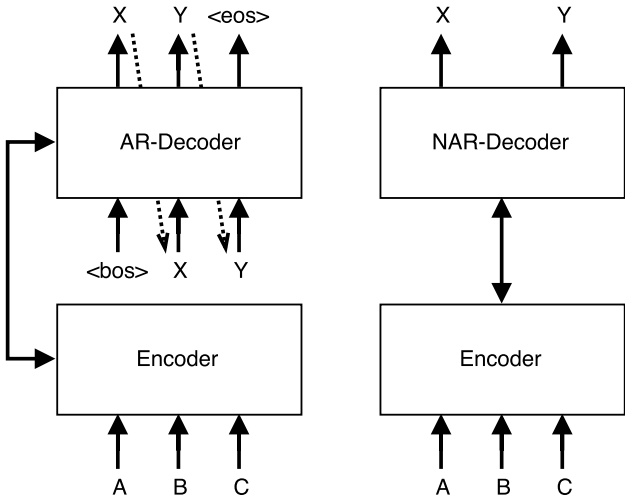


Figure 8.1: Translating "A B C" to "X Y" using autoregressive and non-autoregressive neural MT architectures. The latter generates all output tokens in parallel.

this parallelism during training. One such approach replaces recurrent layers in the decoder with masked convolution layers (Kalchbrenner et al., 2016; Gehring et al., 2017) that provide the causal structure required by the autoregressive factorization.

A recently introduced option which reduces sequential computation still further is to construct the decoder layers out of self-attention computations that have been causally masked in an analogous way. The state-of-the-art Transformer network takes this approach, which allows information to flow in the decoder across arbitrarily long distances in a constant number of operations, asymptotically fewer than required by convolutional architectures (Vaswani et al., 2017).

8.2.2 Non-Autoregressive Decoding

Pros and cons of autoregressive decoding The autoregressive factorization used by conventional NMT models has several benefits. It corresponds to the word-by-word nature of human language production and effectively captures the distribution of real translations. Autoregressive models achieve state-of-the-art performance on large-scale corpora and are easy to train, while beam search provides an effective local search method for finding approximately-optimal output translations.

But there are also drawbacks. As the individual steps of the decoder must be run sequentially rather than in parallel, autoregressive decoding prevents architectures like the Transformer from fully realizing their train-time performance advantage during inference. Meanwhile, beam search suffers from diminishing returns with respect to beam size (Koehn and Knowles, 2017) and exhibits limited search parallelism because it introduces computational dependence between beams.

Towards non-autoregressive decoding A naïve solution is to remove the autoregressive connection directly from an existing encoder-decoder model. Assuming that the target sequence length T can be modeled with a separate conditional distribution p_L, this becomes

$$p_{\mathcal{NA}}(Y|X;\theta) = p_L(T|x_{1:T'};\theta) \cdot \prod_{t=1}^{T} p(y_t|x_{1:T'};\theta). \tag{8.3}$$

This model still has an explicit likelihood function, and it can still be trained using independent cross-entropy losses on each output distribution. Now, however, these distributions can be computed in parallel at inference time.

8.2.3 The Multimodality Problem

However, this naïve approach does not yield good results, because such a model exhibits complete *conditional independence*. Each token's distribution $p(y_t)$ depends only on the source sentence X. This makes it a poor approximation to the true target distribution, which exhibits strong correlation across time. Intuitively, such a decoder is akin to a panel of human translators each asked to provide a single word of a translation independently of the words their colleagues choose.

In particular, consider an English source sentence like "Thank you." This can be accurately translated into German as any one of "Danke.", "Danke schön.", or "Vielen Dank.", all of which may occur in a given training corpus. This target distribution cannot be represented as a product of independent probability distributions for each of the first, second, and third words, because a conditionally independent distribution cannot allow "Danke schön." and "Vielen Dank." without also licensing "Danke Dank." and "Vielen schön."

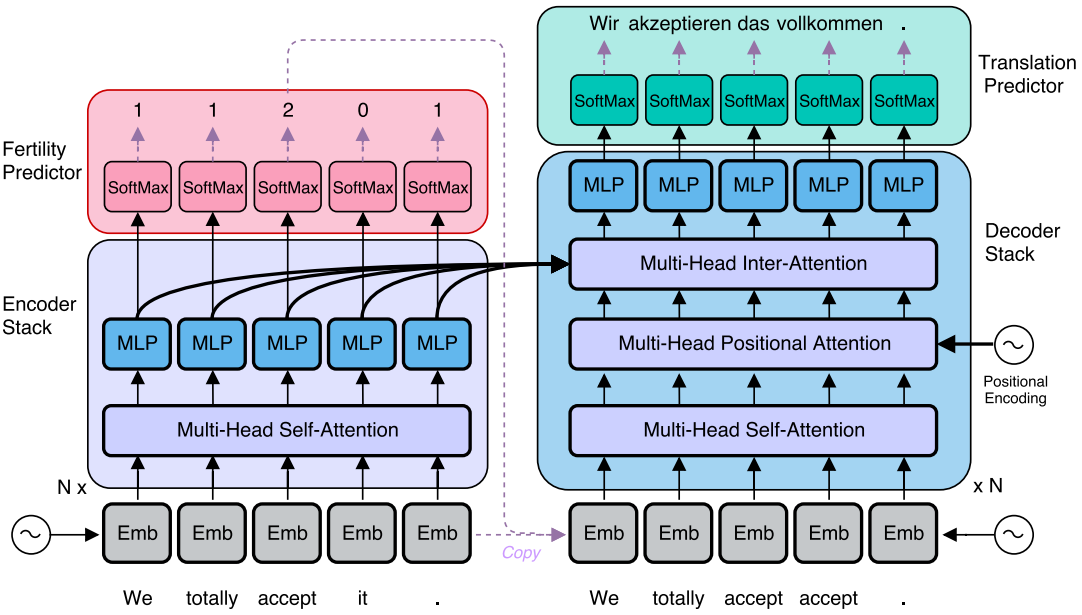


Figure 8.2: The architecture of the NAT, where the black solid arrows represent differentiable connections and the purple dashed arrows are non-differentiable operations. Each sublayer inside the encoder and decoder stacks also includes layer normalization and a residual connection.

The conditional independence assumption prevents a model from properly capturing the highly multimodal distribution of target translations. We call this the "multimodality problem" and introduce both a modified model and new training techniques to tackle this issue.

8.3 The Non-Autoregressive Transformer (NAT)

We introduce a novel NMT model—the Non-Autoregressive Transformer (NAT)—that can produce an entire output translation in parallel. As shown in Fig. 8.2, the model is composed of the following four modules: an encoder stack, a decoder stack, a newly added fertility predictor (details in 8.3.3), and a translation predictor for token decoding.

8.3.1 Encoder Stack

Similar to the autoregressive Transformer, both the encoder and decoder stacks are composed entirely of feed-forward networks (MLPs) and multi-head attention modules. Since

no RNNs are used, there is no inherent requirement for sequential execution, making non-autoregressive decoding possible. For our proposed NAT, the encoder stays unchanged from the original Transformer network.

8.3.2 Decoder Stack

In order to translate non-autoregressively and parallelize the decoding process, we modify the decoder stack as follows.

Decoder Inputs Before decoding starts, the NAT needs to know how long the target sentence will be in order to generate all words in parallel. More crucially, we cannot use time-shifted target outputs (during training) or previously predicted outputs (during inference) as the inputs to the first decoder layer. Omitting inputs to the first decoder layer entirely, or using only positional embeddings, resulted in very poor performance. Instead, we initialize the decoding process using copied source inputs from the encoder side. As the source and target sentences are often of different lengths, we propose two methods:

- **Copy source inputs uniformly**: Each decoder input t is a copy of the Round$(T't/T)$-th encoder input. This is equivalent to "scanning" source inputs from left to right with a constant "speed," and results in a decoding process that is deterministic given a (predicted) target length.

- **Copy source inputs using fertilities**: A more powerful way, depicted in Fig. 8.2 and discussed in more detail below, is to copy each encoder input as a decoder input zero or more times, with the number of times each input is copied referred to as that input word's "fertility." In this case the source inputs are scanned from left to right at a "speed" that varies inversely with the fertility of each input; the decoding process is now conditioned on the sequence of fertilities, while the resulting output length is determined by the sum of all fertility values.

Non-causal self-attention Without the constraint of an autoregressive factorization of the output distribution, we no longer need to prevent earlier decoding steps from accessing information from later steps. Thus we can avoid the causal mask used in the self-attention

module of the conventional Transformer's decoder. Instead, we mask out each query position only from attending to itself, which we found to improve decoder performance relative to unmasked self-attention.

Positional attention We also include an additional positional attention module in each decoder layer, which is a multi-head attention module with the same general attention mechanism used in other parts of the Transformer network, i.e.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{\text{model}}}}\right) \cdot V, \tag{8.4}$$

where d_{model} is the model hidden size, but with the positional encoding[1] as both query and key and the decoder states as the value. This incorporates positional information directly into the attention process and provides a stronger positional signal than the embedding layer alone. We also hypothesize that this additional information improves the decoder's ability to perform local reordering.

8.3.3 Modeling Fertility to Tackle the Multimodality Problem

The multimodality problem can be attacked by introducing a latent variable z to directly model the nondeterminism in the translation process: we first sample z from a prior distribution and then condition on z to non-autoregressively generate a translation.

One way to interpret this latent variable is as a sentence-level "plan" akin to those discussed in the language production literature (Martin et al., 2010). There are several desirable properties for this latent variable:

- It should be simple to infer a value for the latent variable given a particular input-output pair, as this is needed to train the model end-to-end.
- Adding z to the conditioning context should account as much as possible for the correlations across time between different outputs, so that the remaining marginal probabilities at each output location are as close as possible to satisfying conditional independence.

[1]The positional encoding p is computed as $p(j,k) = \sin\left(j/10000^{k/d}\right)$ (for even k) or $\cos\left(j/10000^{k/d}\right)$ (for odd k), where j is the timestep index and k is the channel index.

- It should not account for the variation in output translations so directly that $p(y|x, z)$ becomes trivial to learn, since that is the function our decoder neural network will approximate.

The factorization by length introduced in Eq. 8.3 provides a very weak example of a latent variable model, satisfying the first and third property but not the first. We propose the use of *fertilities* instead. These are integers for each word in the source sentence that correspond to the number of words in the target sentence that can be aligned to that source word using a hard alignment algorithm like IBM Model 2 (Brown et al., 1993).

One of the most important properties of the proposed NAT is that it naturally introduces an informative latent variable when we choose to copy the encoder inputs based on predicted fertilities. More precisely, given a source sentence X, the conditional probability of a target translation Y is:

$$p_{\mathcal{NA}}(Y|X;\theta) = \sum_{f_1,...,f_{T'}\in\mathcal{F}} \left(\prod_{t'=1}^{T'} p_F(f_{t'}|x_{1:T'};\theta) \cdot \prod_{t=1}^{T} p(y_t|x_1\{f_1\},..,x_{T'}\{f_{T'}\};\theta) \right) \quad (8.5)$$

where $\mathcal{F} = \{f_1, ..., f_{T'} | \sum_{t'=1}^{T'} f_{t'} = T, f_{t'} \in \mathbb{Z}^*\}$ is the set of all fertility sequences—one fertility value per source word—that sum to the length of Y and $x\{f\}$ denotes the token x repeated f times.

Fertility prediction As shown in Fig. 8.2, we model the fertility $p_F(f_{t'}|x_{1:T'})$ at each position independently using a one-layer neural network with a softmax classifier ($L = 50$ in our experiments) on top of the output of the last encoder layer. This models the way that fertility values are a property of each input word but depend on information and context from the entire sentence.

Benefits of fertility Fertilities possess all three of the properties listed earlier as desired of a latent variable for non-autoregressive machine translation:

- An external aligner provides a simple and fast approximate inference model that effectively reduces the unsupervised training problem to two supervised ones.

- Using fertilities as a latent variable makes significant progress towards solving the multi-modality problem by providing a natural factorization of the output space. Given a source sentence, restricting the output distribution to those target sentences consistent with a particular fertility sequence dramatically reduces the mode space. Furthermore, the global choice of mode is factored into a set of local mode choices: namely, how to translate each input word. These local mode choices can be effectively supervised because the fertilities provide a fixed "scaffold."

- Including both fertilities and reordering in the latent variable would provide complete alignment statistics. This would make the decoding function trivially easy to approximate given the latent variable and force all of the modeling complexity into the encoder. Using fertilities alone allows the decoder to take some of this burden off of the encoder.

Our use of fertilities as a latent variable also means that there is no need to have a separate means of explicitly modeling the length of the translation, which is simply the sum of fertilities. And fertilities provide a powerful way to condition the decoding process, allowing the model to generate diverse translations by sampling over the fertility space.

8.3.4 Translation Predictor and the Decoding Process

At inference time, the model can identify the translation with the highest conditional probability (see Eq. 8.5) by marginalizing over all possible latent fertility sequences. Given a fertility sequence, however, identifying the optimal translation only requires independently maximizing the local probability for each output position. We define $Y = G(x_{1:T'}, f_{1:T'}; \theta)$ to represent the optimal translation given a source sentence and a sequence of fertility values.

But searching and marginalizing over the whole fertility space is still intractable. We propose three heuristic decoding algorithms to reduce the search space of the NAT model:

Argmax decoding Since the fertility sequence is also modeled with a conditionally independent factorization, we can simply estimate the best translation by choosing the highest-probability fertility for each input word:

$$\hat{Y}_{\text{argmax}} = G(x_{1:T'}, \hat{f}_{1:T'}; \theta), \text{where } \hat{f}_{t'} = \underset{f}{\text{argmax}}\, p_F(f_{t'}|x_{1:T'}; \theta) \tag{8.6}$$

Average decoding We can also estimate each fertility as the expectation of its corresponding softmax distribution:

$$\hat{Y}_{\text{average}} = G(x_{1:T'}, \hat{f}_{1:T'}; \theta), \text{where } \hat{f}_{t'} = \text{Round}\left(\sum_{f_{t'}=1}^{L} p_F(f_{t'}|x_{1:T'}; \theta) f_{t'}\right) \tag{8.7}$$

Noisy parallel decoding (NPD) A more accurate approximation of the true optimum of the target distribution, inspired by Cho (2016), is to draw samples from the fertility space and compute the best translation for each fertility sequence. We can then use the autoregressive teacher to identify the best overall translation:

$$\hat{Y}_{\text{NPD}} = G(x_{1:T'}, \underset{f_{t'} \sim p_F}{\text{argmax}}\, p_{\mathcal{AR}}(G(x_{1:T'}, f_{1:T'}; \theta)|X; \theta); \theta) \tag{8.8}$$

Note that, when using an autoregressive model as a scoring function for a set of decoded translations, it can run as fast as it does at train time because it can be provided with all decoder inputs in parallel.

NPD is a stochastic search method, and it also increases the computational resources required linearly by the sample size. However, because all the search samples can be computed and scored entirely independently, the process only doubles the latency compared to computing a single translation if sufficient parallelism is available.

8.4 Training

The proposed NAT contains a discrete sequential latent variable $f_{1:T'}$, whose conditional posterior distribution $p(f_{1:T'}|x_{1:T'}, y_{1:T}; \theta)$ we can approximate using a proposal distribution $q(f_{1:T'}|x_{1:T'}, y_{1:T})$. This provides a variational bound for the overall maximum likelihood loss:

$$\begin{aligned}\mathcal{L}_{\text{ML}} &= \log p_{\mathcal{NA}}(Y|X;\theta) = \log \sum_{f_{1:T'} \in \mathcal{F}} p_F(f_{1:T'}|x_{1:T'};\theta) \cdot p(y_{1:T}|x_{1:T'}, f_{1:T'};\theta) \\ &\geq \mathop{\mathbb{E}}_{f_{1:T'} \sim q} \left(\underbrace{\sum_{t=1}^{T} \log p(y_t|x_1\{f_1\}, .., x_{T'}\{f_{T'}\};\theta)}_{\text{Translation Loss}} + \underbrace{\sum_{t'=1}^{T'} \log p_F(f_{t'}|x_{1:T'};\theta)}_{\text{Fertility Loss}} \right) + \mathcal{H}(q)\end{aligned} \tag{8.9}$$

We choose a proposal distribution q defined by a separate, fixed fertility model. Possible options include the output of an external aligner, which produces a deterministic sequence of integer fertilities for each (source, target) pair in a training corpus, or fertilities computed from the attention weights used in our fixed autoregressive teacher model. This simplifies the inference process considerably, as the expectation over q is deterministic.

The resulting loss function, consisting of the two bracketed terms in Eq. 8.9, allows us to train the entire model in a supervised fashion, using the inferred fertilities to simultaneously train the translation model p and supervise the fertility neural network model p_F.

8.4.1 Sequence-Level Knowledge Distillation

While the latent fertility model substantially improves the ability of the non-autoregressive output distribution to approximate the multimodal target distribution, it does not completely solve the problem of nondeterminism in the training data. In many cases, there are multiple correct translations consistent with a single sequence of fertilities—for instance, both "Danke schön." and "Vielen dank." are consistent with the English input "Thank you." and the fertility sequence $[2, 0, 1]$, because "you" is not directly translated in either German sentence.

Thus we additionally apply sequence-level knowledge distillation (Kim and Rush, 2016) to construct a new corpus by training an autoregressive machine translation model, known as the teacher, on an existing training corpus, then using that model's greedy outputs as the targets for training the non-autoregressive student. The resulting targets are less noisy and more deterministic, as the trained model will consistently translate a sentence like "Thank you." into the same German translation every time; on the other hand, they are also lower in quality than the original dataset.

8.4.2 Fine-Tuning

Our supervised fertility model enables a decomposition of the overall maximum likelihood loss into translation and fertility terms, but it has some drawbacks compared to variational training. In particular, it heavily relies on the deterministic, approximate inference model provided by the external alignment system, while it would be desirable to train the entire model, including the fertility predictor, end to end.

Thus we propose a fine-tuning step after training the NAT to convergence. We introduce an additional loss term consisting of the reverse K-L divergence with the teacher output distribution, a form of word-level knowledge distillation:

$$\mathcal{L}_{\text{RKL}}\left(f_{1:T'};\theta\right) = \sum_{t=1}^{T}\sum_{y_t}\left[\log p_{\mathcal{AR}}\left(y_t|\hat{y}_{1:t-1},x_{1:T'}\right)\cdot p_{\mathcal{NA}}\left(y_t|x_{1:T'},f_{1:T'};\theta\right)\right], \tag{8.10}$$

where $\hat{y}_{1:T} = G(x_{1:T'}, f_{1:T'};\theta)$. Such a loss is more favorable towards highly peaked student output distributions than a standard cross-entropy error would be.

Then we train the whole model jointly with a weighted sum of the original distillation loss and two such terms, one an expectation over the predicted fertility distribution, normalized with a baseline, and the other based on the external fertility inference model:

$$\mathcal{L}_{\text{FT}} = \lambda\left(\underbrace{\mathop{\mathbb{E}}_{f_{1:T'}\sim p_F}\left(\mathcal{L}_{\text{RKL}}\left(f_{1:T'}\right)-\mathcal{L}_{\text{RKL}}\left(\bar{f}_{1:T'}\right)\right)}_{\mathcal{L}_{\text{RL}}}+\underbrace{\mathop{\mathbb{E}}_{f_{1:T'}\sim q}\left(\mathcal{L}_{\text{RKL}}\left(f_{1:T'}\right)\right)}_{\mathcal{L}_{\text{BP}}}\right)+(1-\lambda)\mathcal{L}_{\text{KD}} \tag{8.11}$$

where $\bar{f}_{1:T'}$ is the average fertility computed by Eq. 8.7. The gradient with respect to the non-differentiable $\mathcal{L}_{\text{RL}}$ term can be estimated with REINFORCE (Williams, 1992), while the $\mathcal{L}_{\text{BP}}$ term can be trained using ordinary backpropagation.

8.5 Experiments

8.5.1 Experimental Settings

Dataset We evaluate the proposed NAT on three widely used public machine translation corpora: IWSLT16 En–De[2], WMT14 En–De,[3] and WMT16 En–Ro[4]. We use IWSLT—which is smaller than the other two datasets—as the development dataset for ablation experiments, and additionally train and test our primary models on both directions of both WMT datasets. All the data are tokenized and segmented into subword symbols using byte-pair encoding (BPE) (Sennrich et al., 2015b) to restrict the size of the vocabulary. For both WMT datasets, we use shared BPE vocabulary and additionally share encoder and decoder word embeddings; for IWSLT, we use separate English and German vocabulary and embeddings.

Teacher Sequence-level knowledge distillation is applied to alleviate multimodality in the training dataset, using autoregressive models as the teachers. The same teacher model used for distillation is also used as a scoring function for fine-tuning and noisy parallel decoding.

To enable a fair comparison, and benefit from its high translation quality, we implemented the autoregressive teachers using the state-of-the-art Transformer architecture. In addition, we use the same sizes and hyperparameters for each student and its respective teacher, with the exception of the newly added positional self-attention and fertility prediction modules.

[2]https://wit3.fbk.eu/
[3]http://www.statmt.org/wmt14/translation-task
[4]http://www.statmt.org/wmt16/translation-task

Preparation for knowledge distillation We first train all teacher models using maximum likelihood, then freeze their parameters. To avoid the redundancy of running fixed teacher models repeatedly on the same data, we decode the entire training set once using each teacher to create a new training dataset for its respective student.

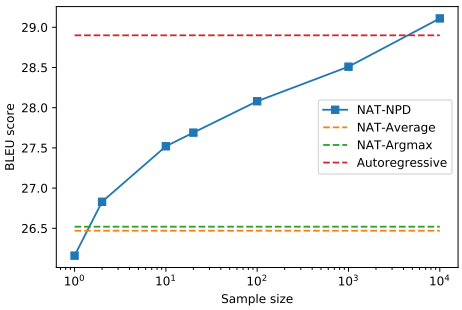


Figure 8.3: BLEU scores on IWSLT development set as a function of sample size for noisy parallel decoding. NPD matches the performance of the other two decoding strategies after two samples, and exceeds the performance of the autoregressive teacher with around 1000.

Encoder initialization We find it helpful to initialize the weights in the NAT student's encoder with the encoder weights from its teacher, as the autoregressive and non-autoregressive models share the same encoder input and architecture.

Fertility supervision during training As described above, we supervise the fertility predictions at train time by using a fixed aligner as a fertility inference function. We use the `fast_align`[5] implementation of IBM Model 2 for this purpose, with default parameters (Dyer et al., 2013).

Hyperparameters For experiments on WMT datasets, we use the hyperparameter settings of the `base` Transformer model described in Vaswani et al. (2017), though without label smoothing. As IWSLT is a smaller corpus, and to reduce training time, we use a set of smaller hyperparameters ($d_{\text{model}} = 287, d_{\text{hidden}} = 507, n_{\text{layer}} = 5, n_{\text{head}} = 2$, and $t_{\text{warmup}} = 746$) for all experiments on that dataset. For fine-tuning we use $\lambda = 0.25$.

Evaluation metrics We evaluate using tokenized and cased BLEU scores (Papineni et al., 2002).

Implementation We have open-sourced our PyTorch implementation of the NAT[6].

[5] https://github.com/clab/fast_align
[6] https://github.com/salesforce/nonauto-nmt

Models	WMT14		WMT16		IWSLT16		
	En→De	De→En	En→Ro	Ro→En	En→De	Latency / Speedup	
NAT	17.35	20.62	26.22	27.83	25.20	39 ms	15.6×
NAT (+FT)	17.69	21.47	27.29	29.06	26.52	39 ms	15.6×
NAT (+FT + NPD $s = 10$)	18.66	22.41	29.02	30.76	27.44	79 ms	7.68×
NAT (+FT + NPD $s = 100$)	19.17	23.20	29.79	**31.44**	28.16	257 ms	2.36×
Autoregressive ($b = 1$)	22.71	26.39	31.35	31.03	28.89	408 ms	1.49×
Autoregressive ($b = 4$)	23.45	27.02	31.91	31.76	29.70	607 ms	1.00×

Table 8.1: BLEU scores on official test sets (`newstest2014` for WMT En-De and `newstest2016` for WMT En-Ro) or the development set for IWSLT. NAT models without NPD use argmax decoding. Latency is computed as the time to decode a single sentence without minibatching, averaged over the whole test set; decoding is implemented in PyTorch on a single NVIDIA Tesla P100.

8.5.2 Results

Across the three datasets we used, the NAT performs between 2-5 BLEU points worse than its autoregressive teacher, with part or all of this gap addressed by the use of noisy parallel decoding. In the case of WMT16 English–Romanian, NPD improves the performance of our non-autoregressive model to within 0.2 BLEU points of the previous overall state of the art (Gehring et al., 2017).

Comparing latencies on the development model shows a speedup of more than a factor of 10 over greedy autoregressive decoding, or a factor of 15 over beam search. Latencies for decoding with NPD, regardless of sample size, could be reduced to about 80ms by parallelizing across multiple GPUs because each sample can be generated, then scored, independently from the others.

8.5.3 Ablation Study

We also conduct an extensive ablation study with the proposed NAT on the IWSLT dataset. First, we note that the model fails to train when provided with only positional embeddings as input to the decoder. Second, we see that training on the distillation corpus rather than the ground truth provides a fairly consistent improvement of around 5 BLEU points. Third,

switching from uniform copying of source inputs to fertility-based copying improves performance by four BLEU points when using ground-truth training or two when using distillation.

Distillation		Decoder Inputs		+PosAtt	Fine-tuning			BLEU	BLEU (T)
b=1	b=4	+uniform	+fertility		$+\mathcal{L}_{\text{KD}}$	$+\mathcal{L}_{\text{BP}}$	$+\mathcal{L}_{\text{RL}}$		
				✓				≈ 2	
		✓		✓				16.51	
			✓	✓				18.87	
✓		✓		✓				20.72	
	✓	✓		✓				21.12	
✓			✓					24.02	43.91
✓			✓	✓				25.20	45.41
✓		✓		✓	✓	✓		22.44	
✓			✓	✓			✓	$\times$	$\times$
✓			✓	✓		✓		$\times$	$\times$
✓			✓	✓	✓	✓		25.76	46.11
✓			✓	✓	✓	✓	✓	**26.52**	**47.38**

Table 8.2: Ablation performance on the IWSLT development set. BLEU (T) refers to the BLEU score on a version of the development set that has been translated by the teacher model. An $\times$ indicates that fine-tuning caused that model to get worse. When uniform copying is used as the decoder inputs, the ground-truth target lengths are provided. All models use argmax decoding.

Fine-tuning does not converge with reinforcement learning alone, or with the $\mathcal{L}_{\text{BP}}$ term alone, but use of all three fine-tuning terms together leads to an improvement of around 1.5 BLEU points. Training the student model from a distillation corpus produced using beam search is similar to training from the greedily-distilled corpus.

We include two examples of translations from the IWSLT development set in Fig. 8.4. Instances of repeated words or phrases, highlighted in gray, are most prevalent in the non-autoregressive output for the relatively complex first example sentence. Two pairs of repeated words in the first example, as well as a pair in the second, are not present in the versions with noisy parallel decoding, suggesting that NPD scoring using the teacher model can filter out such mistakes. The translations produced by the NAT with NPD, while of a similar quality to those produced by the autoregressive model, are also noticeably more

Source:	politicians try to pick words and use words to shape reality and control reality , but in fact , reality changes words far more than words can ever change reality .
Target:	Politiker versuchen Worte zu benutzen , um die Realität zu formen und die Realität zu kontrollieren , aber tatsächlich verändert die Realität Worte viel mehr , als Worte die Realität jemals verändern könnten .
AR:	Politiker versuchen Wörter zu wählen und Wörter zur Realität zu gestalten und Realität zu steuern , aber in Wirklichkeit verändert sich die Realität viel mehr als Worte , die die Realität verändern können .
NAT:	Politiker versuchen , Wörter wählen und zu verwenden , um Realität zu formen und Realität zu formen , aber tatsächlich ändert Realität Realität viel mehr als Worte die Realität Realität verändern .
NAT+NPD:	Politiker versuchen , Wörter wählen und zu verwenden , um Realität Realität formen und die Realität zu formen , aber tatsächlich ändert die Realität Worte viel mehr als Worte jemals die Realität verändern können .
Source:	I see wheelchairs bought and sold like used cars .
Target:	ich erlebe , dass Rollstühle gekauft und verkauft werden wie Gebrauchtwagen
AR:	ich sehe Rollstühlen , die wie Autos verkauft und verkauft werden .
NAT:	ich sehe , dass Stühle Stühle und verkauft wie Autos verkauft .
NAT+NPD:	ich sehe Rollühle kauften und verkaufte wie Autos .

Figure 8.4: Two examples comparing translations produced by an autoregressive (AR) and non-autoregressive Transformer as well as the result of noisy parallel decoding with sample size 100. Repeated words are highlighted in gray.

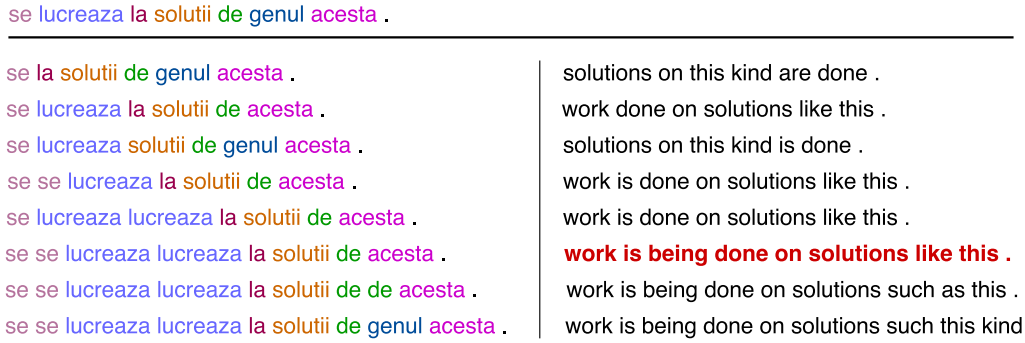


Figure 8.5: A Romanian–English example translated with noisy parallel decoding. At left are eight sampled fertility sequences from the encoder, represented with their corresponding decoder input sequences. Each of these values for the latent variable leads to a different possible output translation, shown at right. The autoregressive Transformer then picks the best translation, shown in red, a process which is much faster than directly using it to generate output.

literal.

We also show an example of the noisy parallel decoding process in Fig. 8.5, demonstrating the diversity of translations that can be found by sampling from the fertility space.

8.6 Schematic and Analysis

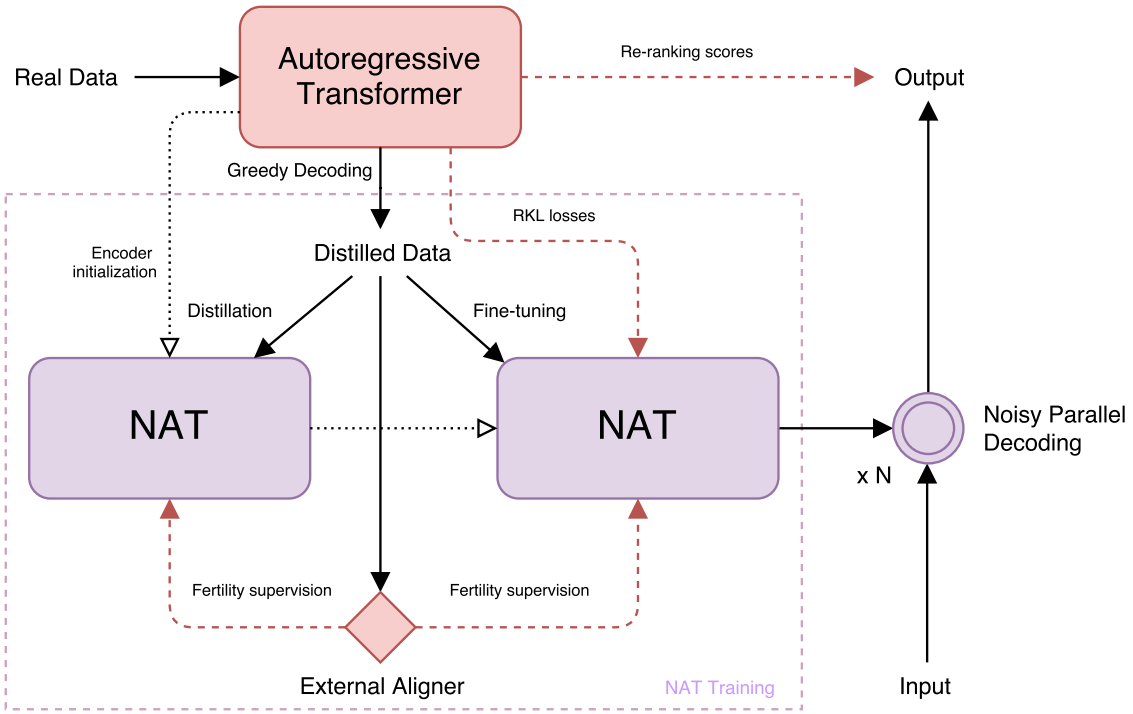


Figure 8.6: The schematic structure of training and inference for the NAT. The "distilled data" contains target sentences decoded by the autoregressive model and ground-truth source sentences.

8.7 Conclusion

We introduce a latent variable model for non-autoregressive machine translation that enables a decoder based on Vaswani et al. (2017) to take full advantage of its exceptional degree of internal parallelism even at inference time. As a result, we measure translation

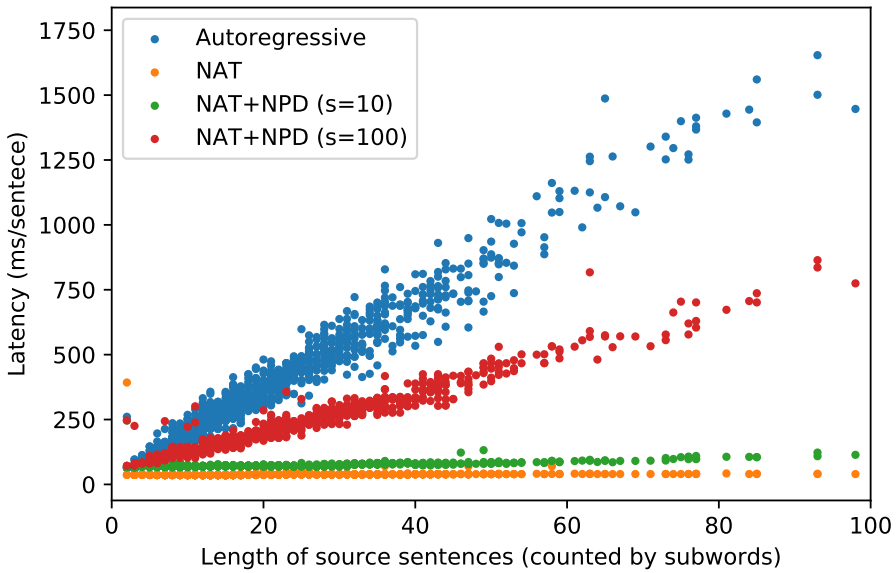


Figure 8.7: The translation latency, computed as the time to decode a single sentence without minibatching, for each sentence in the IWSLT development set as a function of its length. The autoregressive model has latency linear in the decoding length, while the latency of the NAT is nearly constant for typical lengths, even with NPD with sample size 10. When using NPD with sample size 100, the level of parallelism is enough to more than saturate the GPU, leading again to linear latencies.

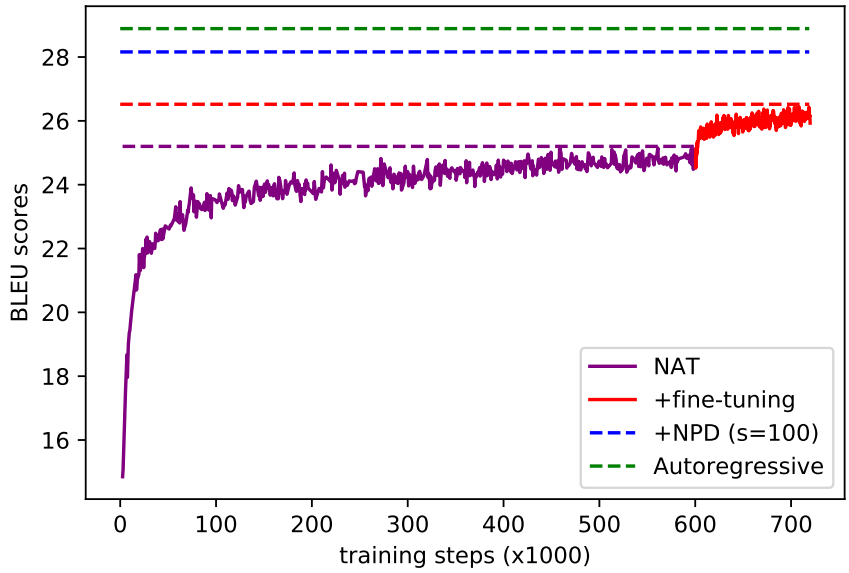


Figure 8.8: Learning curves for training and fine-tuning of the NAT on IWSLT. BLEU scores are on the development set.

latencies of one-tenth that of an equal-sized autoregressive model, while maintaining competitive BLEU scores.

Chapter 9

Real-Time Neural Machine Translation

Simultaneous translation, the task of translating content in real-time as it is produced, is an important tool for real-time understanding of spoken lectures or conversations (Fügen et al., 2007; Bangalore et al., 2012). Different from the typical machine translation (MT) task, in which translation quality is paramount, simultaneous translation requires balancing the trade-off between translation quality and time delay to ensure that users receive translated content in an expeditious manner (Mieno et al., 2015). A number of methods have been proposed to solve this problem, mostly in the context of phrase-based machine translation. These methods are based on a *segmenter*, which receives the input one word at a time, then decides when to send it to a MT system that translates each segment independently (Oda et al., 2014) or with a minimal amount of language model context (Bangalore et al., 2012).

Independently of simultaneous translation, accuracy of standard MT systems has greatly improved with the introduction of neural-network-based MT systems (NMT) (Sutskever et al., 2014; Bahdanau et al., 2014). Very recently, there have been a few efforts to apply NMT to simultaneous translation either through heuristic modifications to the decoding process (Cho and Esipova, 2016), or through the training of an independent segmentation network that chooses when to perform output using a standard NMT model (Satija and Pineau, 2016). However, the former model lacks a capability to learn the appropriate timing with which to perform translation, and the latter model uses a standard NMT model as-is,

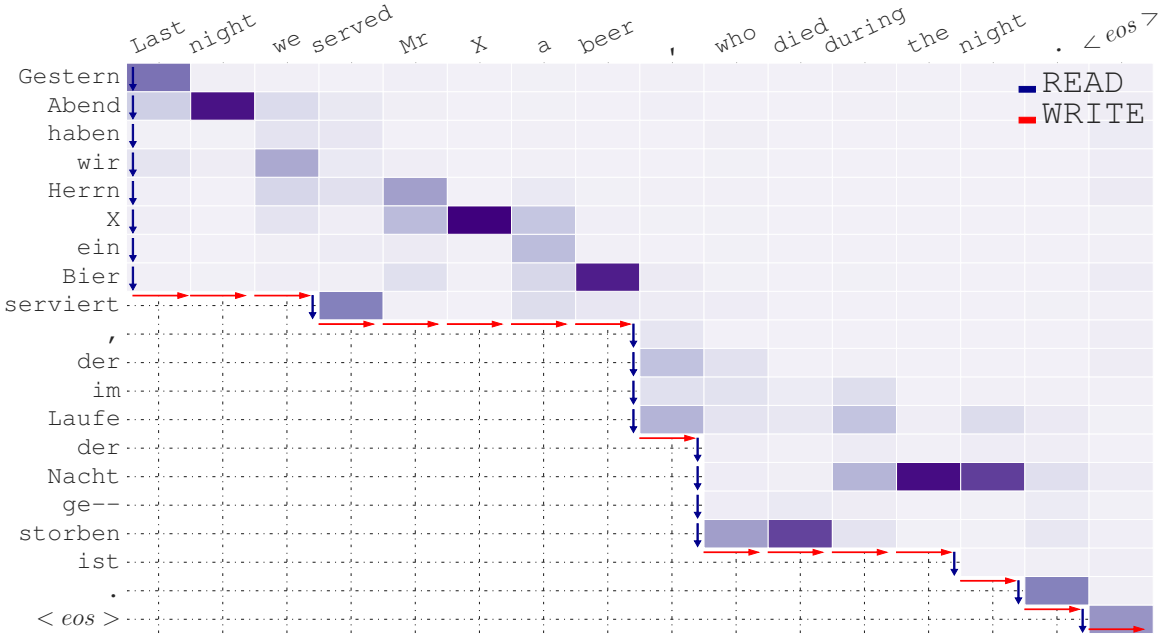


Figure 9.1: Example output from the proposed framework in DE $\rightarrow$ EN simultaneous translation. The heat-map represents the soft alignment between the incoming source sentence (left, up-to-down) and the emitted translation (top, left-to-right). The length of each column represents the number of source words being waited for before emitting the translation. Best viewed when zoomed digitally.

lacking a holistic design of the modeling and learning within the simultaneous MT context. In addition, neither model has demonstrated gains over previous segmentation-based baselines, leaving questions of their relative merit unresolved.

In this paper, we propose a unified design for learning to perform neural simultaneous machine translation. The proposed framework is based on formulating translation as an interleaved sequence of two actions: READ and WRITE. Based on this, we devise a model connecting the NMT system and these READ/WRITE decisions. An example of how translation is performed in this framework is shown in Fig. 9.1, and detailed definitions of the problem and proposed framework are described in §9.1 and §9.2. To learn which actions to take when, we propose a reinforcement-learning-based strategy with a reward function that considers both quality and delay (§9.3). We also develop a beam-search method that performs search within the translation segments (§9.4).

We evaluate the proposed method on English-Russian (EN-RU) and English-German (EN-DE) translation in both directions (§9.5). The quantitative results show strong improvements compared to both the NMT-based algorithm and a conventional segmentation methods. We also extensively analyze the effectiveness of the learning algorithm and the influence of the trade-off in the optimization criterion, by varying a target delay. Finally, qualitative visualization is utilized to discuss the potential and limitations of the framework.

9.1 Problem Definition

Suppose we have a buffer of input words $X = \{x_1, ..., x_{T_s}\}$ to be translated in real-time. We define the simultaneous translation task as sequentially making two interleaved decisions: READ or WRITE. More precisely, the translator READs a source word x_η from the input buffer in chronological order as translation context, or WRITEs a translated word y_τ onto the output buffer, resulting in output sentence $Y = \{y_1, ..., y_{T_t}\}$, and action sequence $A = \{a_1, ..., a_T\}$ consists of T_s READs and T_t WRITEs, so $T = T_s + T_t$.

Similar to standard MT, we have a measure $Q(Y)$ to evaluate the translation quality, such as BLEU score (Papineni et al., 2002). For simultaneous translation we are also concerned with the fact that each action incurs a time delay $D(A)$. $D(A)$ will mainly be influenced by delay caused by READ, as this entails waiting for a human speaker to continue speaking (about 0.3s per word for an average speaker), while WRITE consists of generating a few words from a machine translation system, which is possible on the order of milliseconds. Thus, our objective is finding an optimal policy that generates decision sequences with a good trade-off between higher quality $Q(Y)$ and lower delay $D(A)$. We elaborate on exactly how to define this trade-off in §9.3.2.

In the following sections, we first describe how to connect the READ/WRITE actions with the NMT system (§9.2), and how to optimize the system to improve simultaneous MT results (§9.3).

9.2 Simultaneous Translation with Neural Machine Translation

The proposed framework is shown in Fig. 9.2, and can be naturally decomposed into two parts: environment (§9.2.1) and agent (§9.2.2).

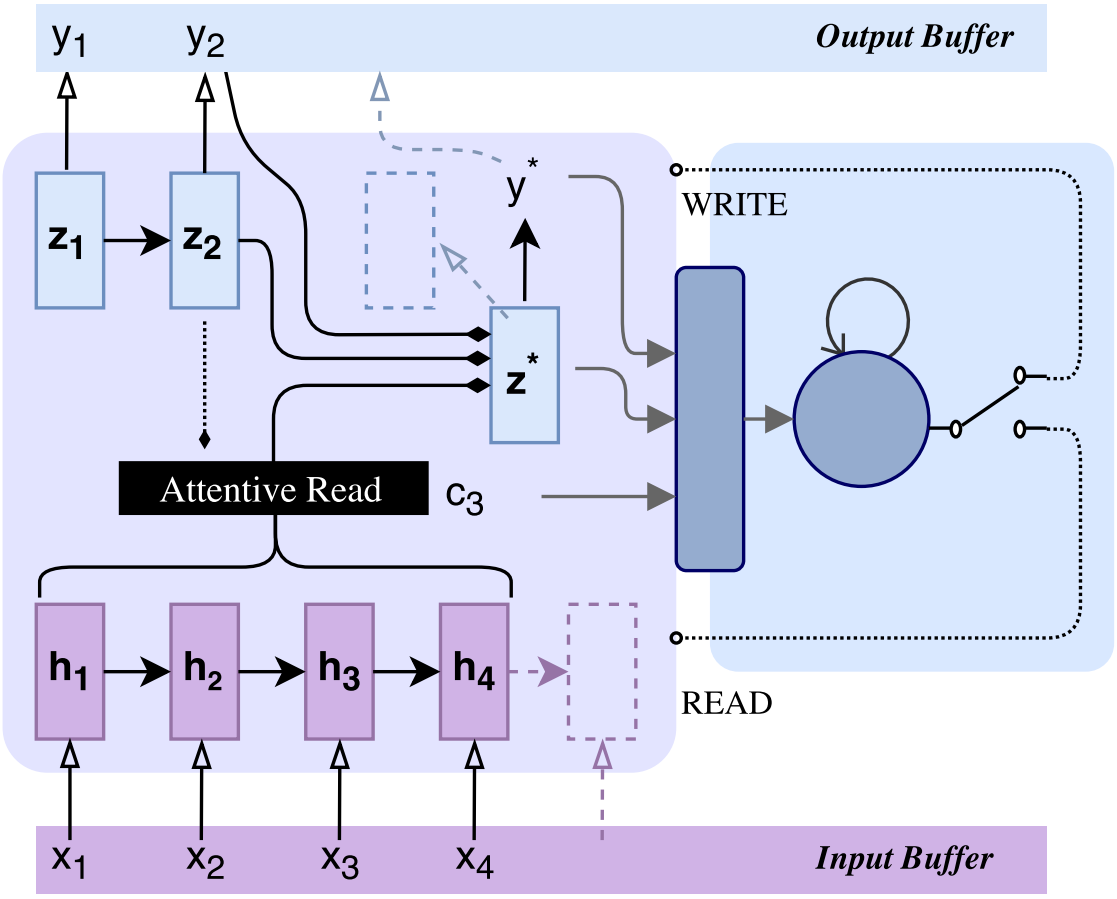


Figure 9.2: Illustration of the proposed framework: at each step, the NMT environment (left) computes a candidate translation. The recurrent agent (right) will the observation including the candidates and send back decisions–READ or WRITE.

9.2.1 Environment

Encoder: READ The first element of the NMT system is the encoder, which converts input words $X = \{x_1, ..., x_{T_s}\}$ into context vectors $H = \{h_1, ..., h_{T_s}\}$. Standard NMT uses bi-directional RNNs as encoders (Bahdanau et al., 2014), but this is not suitable for simultaneous processing as using a reverse-order encoder requires knowing the final word of the sentence before beginning processing. Thus, we utilize a simple left-to-right uni-directional RNN as our encoder:

$$h_\eta = \phi_{\text{UNI-ENC}}\left(h_{\eta-1}, x_\eta\right) \tag{9.1}$$

Decoder: WRITE Similar with standard MT, we use an attention-based decoder. In contrast, we only reference the words that have been read from the input when generating each target word:

$$\begin{aligned} c_\tau^\eta &= \phi_{\text{ATT}}\left(z_{\tau-1}, y_{\tau-1}, H^\eta\right) \\ z_\tau^\eta &= \phi_{\text{DEC}}\left(z_{\tau-1}, y_{\tau-1}, c_\tau^\eta\right) \\ p\left(y|y_{<\tau}, H^\eta\right) &\propto \exp\left[\phi_{\text{OUT}}\left(z_\tau^\eta\right)\right], \end{aligned} \tag{9.2}$$

where for τ, $z_{\tau-1}$ and $y_{\tau-1}$ represent the previous decoder state and output word, respectively. H^η is used to represent the incomplete input states, where H^η is a prefix of H. As the WRITE action calculates the probability of the next word on the fly, we need greedy decoding for each step:

$$y_\tau^\eta = \text{argmax}_y\, p\left(y|y_{<\tau}, H^\eta\right) \tag{9.3}$$

Note that y_τ^η, z_τ^η corresponds to H^η and is the candidate for y_τ, z_τ. The agent described in the next section decides whether to take this candidate or wait for better predictions.

9.2.2 Agent

A trainable agent is designed to make decisions $A = \{a_1, .., a_T\}, a_t \in \mathcal{A}$ sequentially based on observations $O = \{o_1, ..., o_T\}, o_t \in \mathcal{O}$, and then control the translation environment

properly.

Observation As shown in Fig 9.2, we concatenate the current context vector c_τ^η, the current decoder state z_τ^η and the embedding vector of the candidate word y_τ^η as the continuous observation, $o_{\tau+\eta} = [c_\tau^\eta; z_\tau^\eta; E(y_\tau^\eta)]$ to represent the current state.

Action Similarly to prior work (Grissom II et al., 2014), we define the following set of actions:

- **READ**: the agent rejects the candidate and waits to encode the next word from input buffer;
- **WRITE**: the agent accepts the candidate and emits it as the prediction into output buffer;

Policy How the agent chooses the actions based on the observation defines the policy. In our setting, we utilize a stochastic policy π_θ parameterized by a recurrent neural network, that is:

$$\begin{aligned} s_t &= f_\theta\left(s_{t-1}, o_t\right) \\ \pi_\theta(a_t|a_{<t}, o_{\leq t}) &\propto g_\theta\left(s_t\right), \end{aligned} \tag{9.4}$$

where s_t is the internal state of the agent, and is updated recurrently yielding the distribution of the action a_t. Based on the policy of our agent, the overall algorithm of greedy decoding is shown in Algorithm 4, The algorithm outputs the translation result and a sequence of observation-action pairs.

9.3 Learning

The proposed framework can be trained using reinforcement learning. More precisely, we use policy gradient algorithm together with variance reduction and regularization techniques.

Algorithm 4 Simultaneous Greedy Decoding

Require: NMT system ϕ, policy π_θ, τ_{MAX}, input buffer X, output buffer Y, state buffer S.

Init $x_1 \Leftarrow X, h_1 \leftarrow \phi_{\text{ENC}}(x_1), H^1 \leftarrow \{h_1\}$
  $z_0 \leftarrow \phi_{\text{INIT}}(H^1), y_0 \leftarrow \langle s \rangle$
  $\tau \leftarrow 0, \eta \leftarrow 1$
while $\tau < \tau_{\text{MAX}}$ **do**
  $t \leftarrow \tau + \eta$
  $y_\tau^\eta, z_\tau^\eta, o_t \leftarrow \phi(z_{\tau-1}, y_{\tau-1}, H^\eta)$
  $a_t \sim \pi_\theta(a_t; a_{<t}, o_{<t}), S \Leftarrow (o_t, a_t)$
  if $a_t =$ READ and $x_\eta \neq \langle /s \rangle$ **then**
    $x_{\eta+1} \Leftarrow X, h_{\eta+1} \leftarrow \phi_{\text{ENC}}(h_\eta, x_{\eta+1})$
    $H^{\eta+1} \leftarrow H^\eta \cup \{h_{\eta+1}\}, \eta \leftarrow \eta + 1$
    if $|Y| = 0$ **then** $z_0 \leftarrow \phi_{\text{INIT}}(H^\eta)$
  else if $a_t =$ WRITE **then**
    $z_\tau \leftarrow z_\tau^\eta, y_\tau \leftarrow y_\tau^\eta$
    $Y \Leftarrow y_\tau, \tau \leftarrow \tau + 1$
    if $y_\tau = \langle /s \rangle$ **then break**

9.3.1 Pre-training

We need an NMT environment for the agent to explore and use to generate translations. Here, we simply pre-train the NMT encoder-decoder on full sentence pairs with maximum likelihood, and assume the pre-trained model is still able to generate reasonable translations even on incomplete source sentences. Although this is likely sub-optimal, our NMT environment based on uni-directional RNNs can treat incomplete source sentences in a manner similar to shorter source sentences and has the potential to translate them more-or-less correctly.

9.3.2 Reward Function

The policy is learned in order to increase a reward for the translation. At each step the agent will receive a reward signal r_t based on (o_t, a_t). To evaluate a good simultaneous machine translation, a reward must consider both quality and delay.

Quality We evaluate the translation quality using metrics such as BLEU (Papineni et al., 2002). The BLEU score is defined as the weighted geometric average of the modified n-gram precision BLEU^0, multiplied by the brevity penalty BP to punish a short translation. In practice, the vanilla BLEU score is not a good metric at sentence level because being a geometric average, the score will reduce to zero if one of the precisions is zero. To avoid this, we used a smoothed version of BLEU for our implementation (Lin and Och, 2004).

$$\mathrm{BLEU}(Y, Y^*) = \mathrm{BP} \cdot \mathrm{BLEU}^0(Y, Y^*), \tag{9.5}$$

where Y^* is the reference and Y is the output. We decompose BLEU and use the difference of partial BLEU scores as the reward, that is:

$$r_t^Q = \begin{cases} \Delta\mathrm{BLEU}^0(Y, Y^*, t) & t < T \\ \mathrm{BLEU}(Y, Y^*) & t = T \end{cases} \tag{9.6}$$

where Y^t is the cumulative output at t ($Y^0 = \emptyset$), and $\Delta\mathrm{BLEU}^0(Y, Y^*, t) = \mathrm{BLEU}^0(Y^t, Y^*) - \mathrm{BLEU}^0(Y^{t-1}, Y^*)$. Obviously, if $a_t = \textsc{read}$, no new words are written into Y, yielding $r_t^Q = 0$. Note that we do not multiply BP until the end of the sentence, as it would heavily penalize partial translation results.

Delay As another critical feature, delay judges how much time is wasted waiting for the translation. Ideally we would directly measure the actual time delay incurred by waiting for the next word. For simplicity, however, we suppose it consumes the same amount of time listening for one more word. We define two measurements, global and local, respectively:

- **Average Proportion (AP)**: following the definition in (Cho and Esipova, 2016), X, Y are the source and decoded sequences respectively, and we use $s(\tau)$ to denote the number of source words been waited when decoding word y_τ,

$$
\begin{aligned}
0 < d\left(X, Y\right) &= \frac{1}{|X||Y|} \sum_{\tau} s(\tau) \leq 1 \\
d_t &= \begin{cases} 0 & t < T \\ d(X, Y) & t = T \end{cases}
\end{aligned}
\tag{9.7}
$$

d is a global delay metric, which defines the average waiting proportion of the source sentence when translating each word.

- **Consecutive Wait length (CW)**: in speech translation, listeners are also concerned with long silences during which no translation occurs. To capture this, we also consider on how many words were waited for (READ) consecutively between translating two words. For each action, where we initially define $c_0 = 0$,

$$
c_t = \begin{cases} c_{t-1} + 1 & a_t = \text{READ} \\ 0 & a_t = \text{WRITE} \end{cases}
\tag{9.8}
$$

- **Target Delay:** We further define "target delay" for both d and c as d^* and c^*, respectively, as different simultaneous translation applications may have different requirements on delay. In our implementation, the reward function for delay is written as:

$$
r_t^D = \alpha \cdot [\text{sgn}(c_t - c^*) + 1] + \beta \cdot \lfloor d_t - d^* \rfloor_+
\tag{9.9}
$$

where $\alpha \leq 0, \beta \leq 0$.

Trade-off between quality and delay A good simultaneous translation system requires balancing the trade-off of translation quality and time delay. Obviously, achieving the best translation quality and the shortest translation delays are in a sense contradictory. In this paper, the trade-off is achieved by balancing the rewards $r_t = r_t^Q + r_t^D$ provided to the system, that is, by adjusting the coefficients α, β and the target delay d^*, c^* in Eq. 9.9.

9.3.3 Reinforcement Learning

Policy Gradient We freeze the pre-trained parameters of an NMT model, and train the agent using the policy gradient (Williams, 1992). The policy gradient maximizes the following expected cumulative future rewards, $J = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T} r_t\right]$, whose gradient is

$$\nabla_\theta J = \mathbb{E}_{\pi_\theta}\left[\sum_{t'=1}^{T} \nabla_\theta \log \pi_\theta(a_{t'}|\cdot) R_t\right] \tag{9.10}$$

$R_t = \sum_{k=t}^{T}\left[r_k^Q + r_k^D\right]$ is the cumulative future rewards for current observation and action. In practice, Eq. 9.10 is estimated by sampling multiple action trajectories from the current policy π_θ, collecting the corresponding rewards.

Variance Reduction Directly using the policy gradient suffers from high variance, which makes learning unstable and inefficient. We thus employ the variance reduction techniques suggested by (Mnih and Gregor, 2014). We subtract from R_t the output of a baseline network b_φ to obtain $\hat{R}_t = R_t - b_\varphi(o_t)$, and centered re-scale the reward as $\tilde{R}_t = \frac{\hat{R}_t - b}{\sqrt{\sigma^2+\epsilon}}$ with a running average b and standard deviation σ. The baseline network is trained to minimize the squared loss as follows:

$$L_\varphi = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T} \|R_t - b_\varphi(o_t)\|^2\right] \tag{9.11}$$

We also regularize the negative entropy of the policy to facilitate exploration.

The overall learning algorithm is summarized in Algorithm 5. For efficiency, instead of updating with stochastic gradient descent (SGD) on a single sentence, both the agent and the baseline are optimized using a minibatch of multiple sentences.

9.4 Simultaneous Beam Search

In previous sections we described a simultaneous greedy decoding algorithm. In standard NMT it has been shown that beam search, where the decoder keeps a beam of k translation trajectories, greatly improves translation quality (Sutskever et al., 2014), as shown in

Algorithm 5 Learning with Policy Gradient

Require: NMT system ϕ, agent θ, baseline φ
Pretrain the NMT system ϕ using MLE;
Initialize the agent θ;
while stopping criterion fails **do**
    Obtain a translation pairs: $\{(X, Y^*)\}$;
    for $(Y, S) \sim$ Simultaneous Decoding **do**
        for (o_t, a_t) in S **do**
            Compute the quality: r_t^Q;
            Compute the delay: r_t^D;
            Compute the baseline: $b_\varphi(o_t)$;
    Collect the future rewards: $\{R_t\}$;
    Perform variance reduction: $\{\tilde{R}_t\}$;
    Update: $\theta \leftarrow \theta + \lambda_1 \nabla_\theta [J - \kappa \mathcal{H}(\pi_\theta)]$
    Update: $\varphi \leftarrow \varphi - \lambda_2 \nabla_\varphi L$

Fig. 9.3 (A).

It is non-trivial to directly apply beam-search in simultaneous machine translation, as beam search waits until the last word to write down translation. Based on our assumption WRITE does not cost delay, we can perform a simultaneous beam-search when the agent chooses to consecutively WRITE: keep multiple beams of translation trajectories in temporary buffer and output the best path when the agent switches to READ. As shown in Fig. 9.3 (B) & (C), it tries to search for a relatively better path while keeping the delay unchanged.

Note that we do not re-train the agent for simultaneous beam-search. At each step we simply input the observation of the current best trajectory into the agent for making next decision.

9.5 Experiments

9.5.1 Settings

Dataset To extensively study the proposed simultaneous translation model, we train and evaluate it on two different language pairs: "English-German (EN-DE)" and "English-Russian (EN-RU)" in both directions per pair. We use the parallel corpora available from

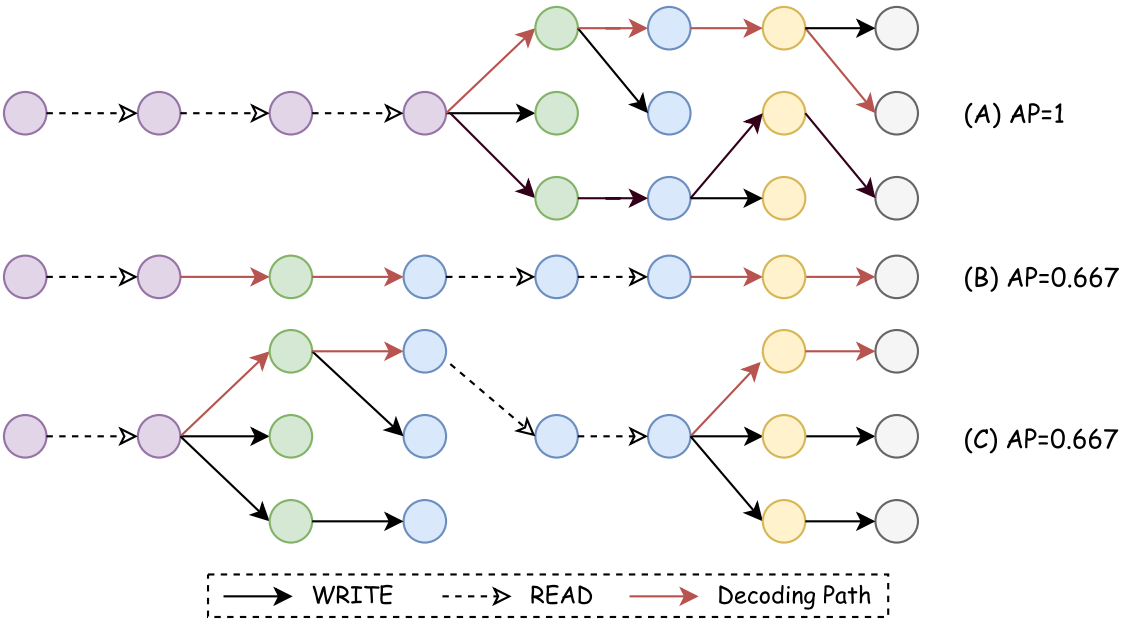
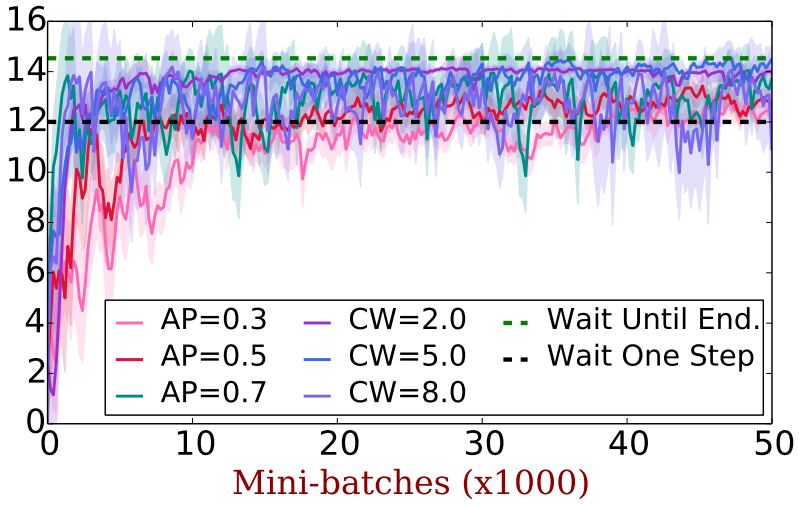


Figure 9.3: Illustrations of (A) beam-search, (B) simultaneous greedy decoding and (C) simultaneous beam-search.

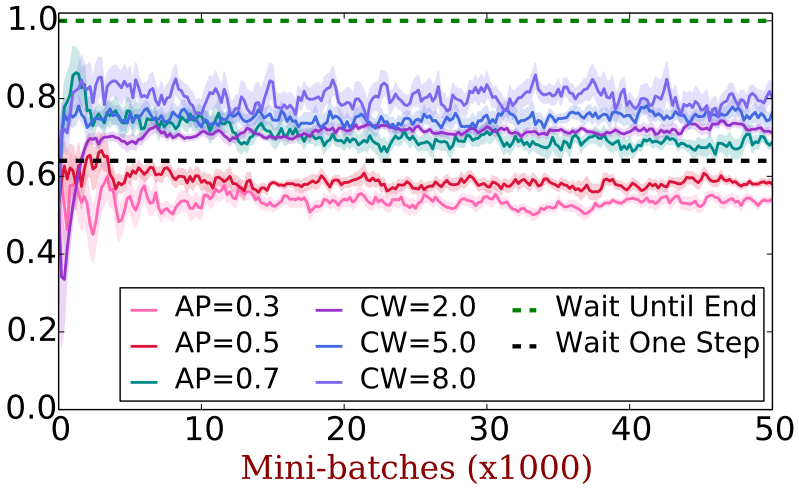
WMT'15[1] for both pre-training the NMT environment and learning the policy. We utilize newstest-2013 as the validation set to evaluate the proposed algorithm. Both the training set and the validation set are tokenized and segmented into sub-word units with byte-pair encoding (BPE) (Sennrich et al., 2015b). We only use sentence pairs where both sides are less than 50 BPE subword symbols long for training.

Environment & Agent Settings We pre-trained the NMT environments for both language pairs and both directions following the same setting from (Cho and Esipova, 2016). We further built our agents, using a recurrent policy with 512 GRUs and a softmax function to produce the action distribution. All our agents are trained using policy gradient using Adam (Kingma and Ba, 2014) optimizer, with a mini-batch size of 10. For each sentence pair in a batch, 5 trajectories are sampled. For testing, instead of sampling we pick the action with higher probability each step.

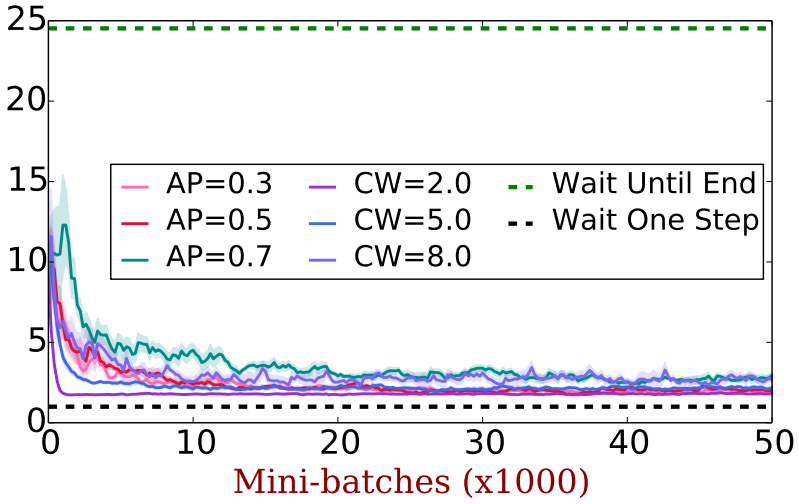
[1]http://www.statmt.org/wmt15/



(a) BLEU (EN → RU)

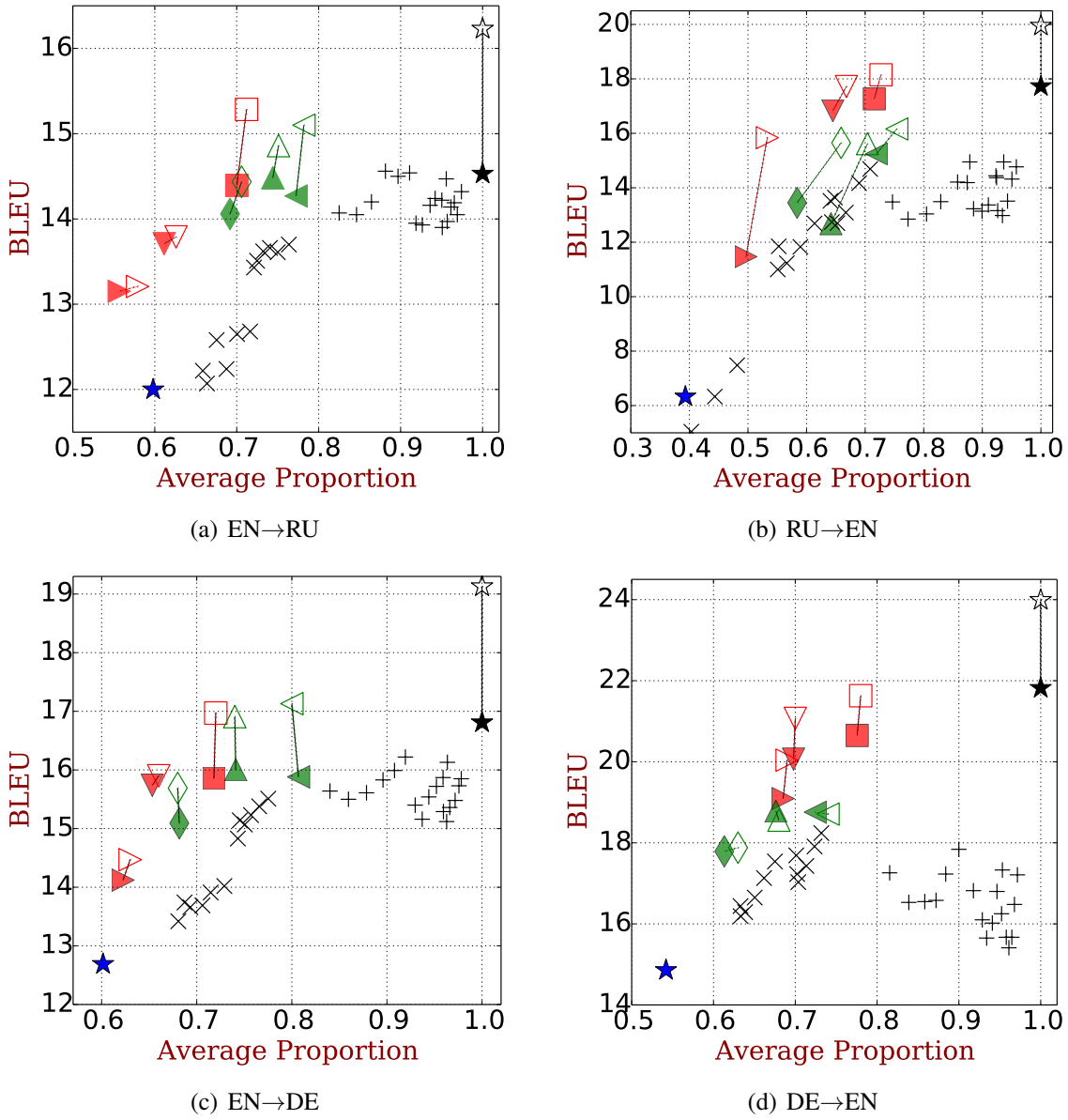


(b) AP (EN → RU)



(c) CW (EN → RU)

Figure 9.4: Learning progress curves for variant delay targets on the validation dataset for



(a) EN→RU

(b) RU→EN

(c) EN→DE

(d) DE→EN

Figure 9.5: Delay (AP) v.s. BLEU for both language pair–directions. The shown point-pairs are the results of simultaneous greedy decoding and beam-search (beam-size = 5) respectively with models trained for various delay targets: (◀ ◁: CW=8, ▲△: CW=5, ◆◇: CW=2, ▶ ▷: AP=0.3, ▼▽: AP=0.5, ■□: AP=0.7). For each target, we select the model that maximizes the quality-to-delay ratio ($\frac{\text{BLEU}}{\text{AP}}$) on the validation set. The baselines are also plotted (★: WOS ★☆: WUE, ×: WID, +: WIW).

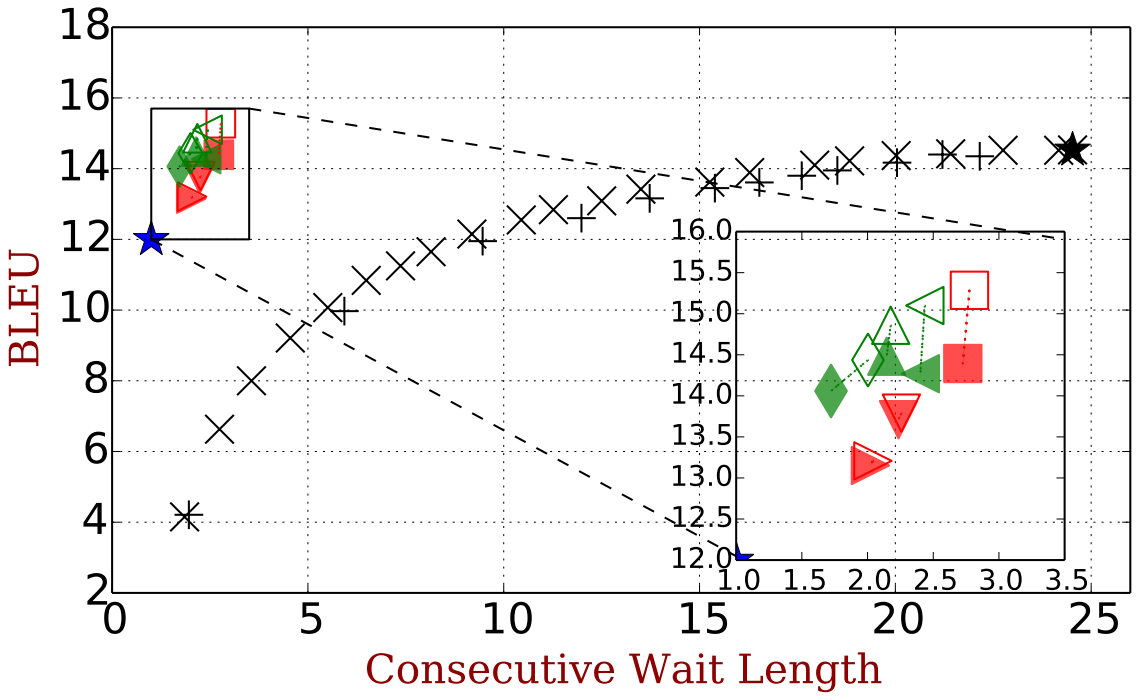


Figure 9.6: Delay (CW) v.s. BLEU score for EN $\rightarrow$ RU, (◀ ◁: CW=8, ▲△: CW=5, ◆◇: CW=2, ▶ ▷: AP=0.3, ▼▽: AP=0.5, ■□: AP=0.7), against the baselines (★: WOS ★: WUE, +: SEG1, $\times$: SEG2).

Baselines We compare the proposed methods against previously proposed baselines. For fair comparison, we use the same NMT environment:

- **Wait-Until-End (WUE)**: an agent that starts to WRITE only when the last source word is seen. In general, we expect this to achieve the best quality of translation. We perform both greedy decoding and beam-search with this method.
- **Wait-One-Step (WOS)**: an agent that WRITEs after each READs. Such a policy is problematic when the source and target language pairs have different word orders or lengths (e.g. EN-DE).
- **Wait-If-Worse/Wait-If-Diff (WIW/WID)**: as proposed by (Cho and Esipova, 2016), the algorithm first pre-READs the next source word, and accepts this READ when the probability of the most likely target word decreases (WIW), or the most likely target word changes (WID).
- **Segmentation-based (SEG)** (Oda et al., 2014): a state-of-the-art segmentation-based algorithm based on optimizing segmentation to achieve the highest quality score. In this paper, we tried the simple greedy method (SEG1) and the greedy method with POS Constraint (SEG2).

9.5.2 Quantitative Analysis

In order to evaluate the effectiveness of our reinforcement learning algorithms with different reward functions, we vary the target delay $d^* \in \{0.3, 0.5, 0.7\}$ and $c^* \in \{2, 5, 8\}$ for Eq. 9.9 separately, and trained agents with α and β adjusted to values that provided stable learning for each language pair according to the validation set.

Learning Curves As shown in Fig. 9.4, we plot learning progress for EN-RU translation with different target settings. It clearly shows that our algorithm effectively increases translation quality for all the models, while pushing the delay close, if not all of the way, to the target value. It can also be noted from Fig. 9.4 (a) and (b) that there exists strong correlation between the two delay measures, implying the agent can learn to decrease both AP and CW simultaneously.

Quality v.s. Delay As shown in Fig. 9.5, it is clear that the trade-off between translation quality and delay has very similar behaviors across both language pairs and directions. The smaller delay (AP or CW) the learning algorithm is targeting, the lower quality (BLEU score) the output translation. It is also interesting to observe that, it is more difficult for "→EN" translation to achieve a lower AP target while maintaining good quality, compared to "EN→". In addition, the models that are optimized on AP tend to perform better than those optimized on CW, especially in "→EN" translation. German and Russian sentences tend to be longer than English, hence require more consecutive waits before being able to emit the next English symbol.

v.s. Baselines In Fig. 9.5 and 9.6, the points closer to the upper left corner achieve better trade-off performance. Compared to WUE and WOS which can ideally achieve the best quality (but the worst delay) and the best delay (but poor quality) respectively, all of our proposed models find a good balance between quality and delay. Some of the proposed models can achieve good BLEU scores close to WUE, while have much smaller delay.

Compared to the method of (Cho and Esipova, 2016) based on two hand-crafted rules (WID, WIW), in most cases our proposed models find better trade-off points, while there are a few exceptions. We also observe that the baseline models have trouble controlling the delay in a reasonable area. In contrast, by optimizing towards a given target delay, our proposed model is stable while maintaining good translation quality.

We also compared against (Oda et al., 2014)'s state-of-the-art segmentation algorithm (SEG). As shown in Fig 9.6, it is clear that although SEG can work with variant segmentation lengths (CW), the proposed model outputs high quality translations at a much smaller CW. We conjecture that this is due to the independence assumption in SEG, while the RNNs and attention mechanism in our model makes it possible to look at the whole history to decide each translated word.

w/o Beam-Search We also plot the results of simultaneous beam-search instead of using greedy decoding. It is clear from Fig. 9.5 and 9.6 that most of the proposed models can achieve an visible increase in quality together with a slight increase in delay. This is because beam-search can help to avoid bad local minima. We also observe that the simultaneous

beam-search cannot bring as much improvement as it did in the standard NMT setting. In most cases, the smaller delay the model achieves, the less beam search can help as it requires longer consecutive WRITE segments for extensive search to be necessary. One possible solution is to consider the beam uncertainty in the agent's READ/WRITE decisions. We leave this to future work.

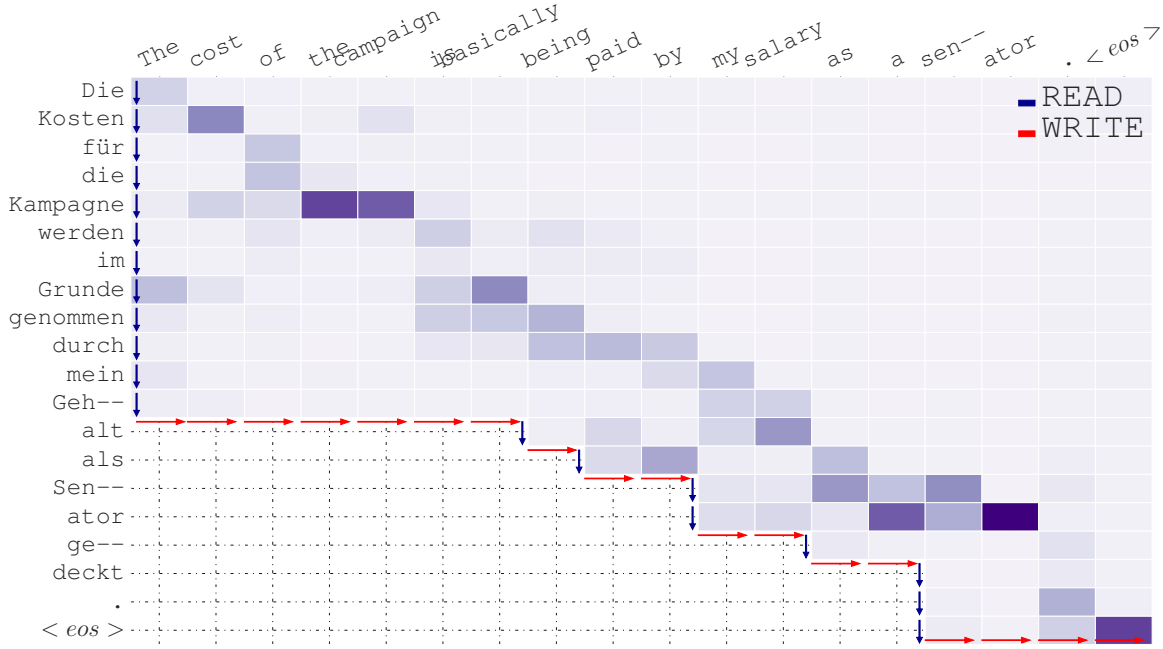
9.5.3 Qualitative Analysis

In this section, we perform a more in-depth analysis using examples from both EN-RU and EN-DE pairs, in order to have a deeper understanding of the proposed algorithm and its remaining limitations. We only perform greedy decoding to simplify visualization.

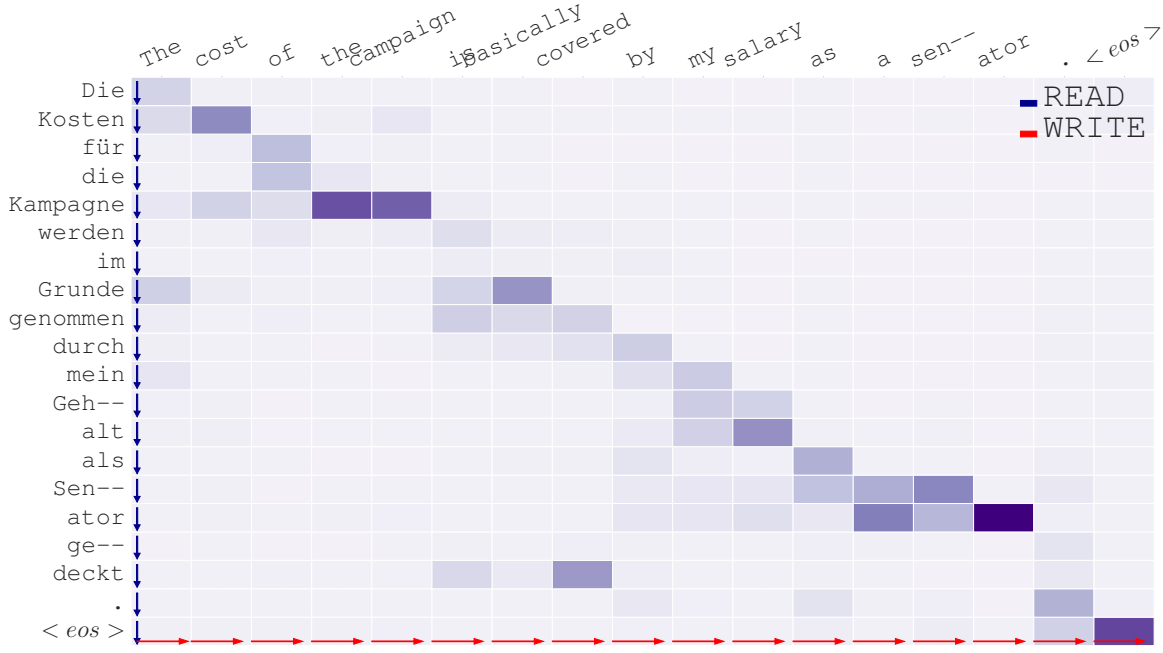
EN→RU As shown in Fig 9.8, since both English and Russian are Subject-–Verb-–Object (SVO) languages, the corresponding words may share the same order in both languages, which makes simultaneous translation easier. It is clear that the larger the target delay (AP or CW) is set, the more words are read before translating the corresponding words, which in turn results in better translation quality. We also note that very early WRITE commonly causes bad translation. For example, for AP=0.3 & CW=2, both the models choose to WRITE in the very beginning the word "The", which is unreasonable since Russian has no articles, and there is no word corresponding to it. One good feature of using NMT is that the more words the decoder READs, the longer history is saved, rendering simultaneous translation easier.

DE→EN As shown in Fig 9.1 and 9.7 (a), where we visualize the attention weights as soft alignment between the progressive input and output sentences, the highest weights are basically along the diagonal line. This indicates that our simultaneous translator works by waiting for enough source words with high alignment weights and then switching to write them.

DE-EN translation is likely more difficult as German usually uses Subject-Object-Verb (SOV) constructions a lot. As shown in Fig 9.1, when a sentence (or a clause) starts the agent has learned such policy to READ multiple steps to approach the verb (e.g. serviert and gestorben in Fig 9.1). Such a policy is still limited when the verb is very far from



(a) Simultaneous Neural Machine Translation



(b) Neural Machine Translation

Figure 9.7: Comparison of DE→EN examples using the proposed framework and usual NMT system respectively. Both the heatmaps share the same setting with Fig. 9.1. The verb "gedeckt" is incorrectly translated in simultaneous translation.

Source	AP=0.3	AP=0.7	CW=2	CW=8
The	The The		The	
people	p-- i-- ent the p-- ol-- s			
,	,		p-- riv--	
as			ers	
I	as I			
heard	я слышал	Люди	,	Люди
in		, как я слышал		
the			как	
countryside	в	в	я слышал	
,				,
want	сельской местности	сельской	в	как я
a		местности		слышал
Government			сельской	
that	,			в
is			местности	сельской
not				
made	хочу		,	
up	правительство			местности
of			хочу	
thi--	,	,		
eves		хотят	правительство	
.				,
<eos>	которое не производится во-- ров .	, чтобы правительство , которое не в-- меши-- вается в во-- ры .	, которое не является состав-- ной частью во-- ров .	хотят , чтобы правительство , которое не в-- меши-- вается в во-- ры .
Summary	BLEU=39/ AP=0.46	BLEU=64/AP=0.77	BLEU=54/CW=1.76	BLEU=64/CW=2.55

Figure 9.8: Given the example input sentence (leftmost column), we show outputs by models trained for various delay targets. For these outputs, each row corresponds to one source word and represents the emitted words (maybe empty) after reading this word. The corresponding source and target words are in the same color for all model outputs.

the subject. For instance in Fig. 9.7, the simultaneous translator achieves almost the same translation with standard NMT except for the verb "gedeckt" which corresponds to "covered" in NMT output. Since there are too many words between the verb "gedeckt" and the subject "Kosten für die Kampagne werden", the agent gives up reading (otherwise it will cause a large delay and a penalty) and WRITEs "being paid" based on the decoder's hypothesis. This is one of the limitations of the proposed framework, as the NMT environment is trained on complete source sentences and it may be difficult to predict the verb that has not been seen in the source sentence. One possible way is to fine-tune the NMT model on incomplete sentences to boost its prediction ability. We will leave this as future work.

9.6 Related Work

Researchers commonly consider the problem of simultaneous machine translation in the scenario of real-time speech interpretation (Fügen et al., 2007; Bangalore et al., 2012; Fujita et al., 2013; Rangarajan Sridhar et al., 2013; Yarmohammadi et al., 2013). In this approach, the incoming speech stream required to be translated are first recognized and segmented based on an automatic speech recognition (ASR) system. The translation model then works independently based on each of these segments, potentially limiting the quality of translation. To avoid using a fixed segmentation algorithm, (Oda et al., 2014) introduced a trainable segmentation component into their system, so that the segmentation leads to better translation quality. (Grissom II et al., 2014) proposed a similar framework, however, based on reinforcement learning. All these methods still rely on translating each segment independently without previous context.

Recently, two research groups have tried to apply the NMT framework to the simultaneous translation task. (Cho and Esipova, 2016) proposed a similar waiting process. However, their waiting criterion is manually defined without learning. (Satija and Pineau, 2016) proposed a method similar to ours in overall concept, but it significantly differs from our proposed method in many details. The biggest difference is that they proposed to use an agent that passively reads a new word at each step. Because of this, it cannot consecutively decode multiple steps, rendering beam search difficult. In addition, they lack the comparison to any existing approaches. On the other hand, we perform an extensive experimental

evaluation against state-of-the-art baselines, demonstrating the relative utility both quantitatively and qualitatively.

The proposed framework is also related to some recent efforts about online sequence-to-sequence (SEQ2SEQ) learning. (Jaitly et al., 2015) proposed a SEQ2SEQ ASR model that takes fixed-sized segments of the input sequence and outputs tokens based on each segment in real-time. It is trained with alignment information using supervised learning. A similar idea for online ASR is proposed by (Luo et al., 2016). Similar to (Satija and Pineau, 2016), they also used reinforcement learning to decide whether to emit a token while reading a new input at each step. Although sharing some similarities, ASR is very different from simultaneous MT with a more intuitive definition for segmentation. In addition, (Yu et al., 2016) recently proposed an online alignment model to help sentence compression and morphological inflection. They regarded the alignment between the input and output sequences as a hidden variable, and performed transitions over the input and output sequence. By contrast, the proposed READ and WRITE actions do not necessarily to be performed on aligned words (e.g. in Fig. 9.1), and are learned to balance the trade-off of quality and delay.

9.7 Conclusion

We propose a unified framework to do neural simultaneous machine translation. To trade off quality and delay, we extensively explore various targets for delay and design a method for beam-search applicable in the simultaneous MT setting. Experiments against state-of-the-art baselines on two language pairs demonstrate the efficacy both quantitatively and qualitatively.

Chapter 10

Conclusion and Future Work

Bibliography

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*. pages 3981–3989.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017a. Learning bilingual word embeddings with (almost) no bilingual data. In *ACL*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017b. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 451–462.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *Proceedings of International Conference on Learning Representations (ICLR)*. Vancouver, Canada.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086* .

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

Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. 2012. Real-time incremental speech-to-speech translation of dialogs. In

Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, pages 437–445.

Yoshua Bengio, Rejean Ducharme, and Pascal Vincent. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3:1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075* .

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics* 19(2):263–311.

Ozan Caglayan, Walid Aransa, Yaxing Wang, Marc Masana, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, and Joost van de Weijer. 2016. Does multimodality help human and machine for translation and image captioning? *arXiv preprint arXiv:1605.09186* .

Chris Callison-Burch, Colin Bannard, and Josh Schroeder. 2005. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 255–262.

Yun Chen, Yang Liu, Yong Cheng, and Victor OK Li. 2017. A teacher-student framework for zero-resource neural machine translation. *arXiv preprint arXiv:1705.00753* .

Yun Chen, Yang Liu, and Victor OK Li. 2018. Zero-resource neural machine translation with multi-agent communication game. *arXiv preprint arXiv:1802.03116* .

Yong Cheng, Yang Liu, Qian Yang, Maosong Sun, and Wei Xu. 2016. Neural machine translation with pivot languages. *arXiv preprint arXiv:1611.04928* .

Kyunghyun Cho. 2015. Natural language understanding with distributed representation. *arXiv preprint arXiv:1511.07916* .

Kyunghyun Cho. 2016. Noisy parallel approximate decoding for conditional recurrent language model. *arXiv preprint arXiv:1605.03835* .

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *arXiv preprint arXiv:1606.02012* .

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–Decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. Nyu-mila neural machine translation systems for wmt16. In *Proceedings of the First Conference on Machine Translation, Berlin, Germany. Association for Computational Linguistics*.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. *arXiv preprint arXiv:1601.01085* .

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *ICLR*.

Marta R Costa-Jussa and José AR Fonollosa. 2016. Character-based neural machine translation. *arXiv preprint arXiv:1603.00810* .

Josep Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, et al. 2016. Systran's pure neural machine translation systems. *arXiv preprint arXiv:1610.05540* .

Jacob Devlin. 2017. Sharp models on dull hardware: Fast and accurate neural machine translation decoding on the cpu. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2810–2815.

Jacob Devlin, Saurabh Gupta, Ross Girshick, Margaret Mitchell, and C Lawrence Zitnick. 2015. Exploring nearest neighbor approaches for image captioning. *arXiv preprint arXiv:1505.04467* .

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. ACL.

Chris Dyer, Victor Chahuneau, and Noah Smith. 2013. A simple, fast, and effective reparameterization of IBM Model 2. In *NAACL*.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. *arXiv preprint arXiv:1702.03525* .

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *NAACL*.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T Yarman Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. In *EMNLP*.

Christian Fügen, Alex Waibel, and Muntsin Kolss. 2007. Simultaneous translation of lectures and speeches. *Machine Translation* 21(4):209–252.

Tomoki Fujita, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2013. Simple, lexicalized choice of translation timing for simultaneous speech translation. In *INTERSPEECH*.

Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2016. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344* .

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of International Conference on Machine Learning (ICML)*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* .

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1342–1352.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018a. Non-autoregressive neural machine translation. *ICLR* .

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor OK Li. 2018b. Universal neural machine translation for extremely low resource languages. *arXiv preprint arXiv:1802.05368* .

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016a. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393* .

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor OK Li. 2016b. Learning to translate in real-time with neural machine translation. *arXiv preprint arXiv:1610.00388* .

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148* .

Caglar Gulcehre, Sarath Chandar, Kyunghyun Cho, and Yoshua Bengio. 2018. Dynamic neural turing machine with continuous and discrete addressing schemes. *Neural computation* 30(4):857–884.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535* .

David Ha, Andrew Dai, and Quoc V Le. 2016a. Hypernetworks. *arXiv preprint arXiv:1609.09106* .

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016b. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint arXiv:1611.04798* .

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. *CoRR* abs/1803.05567.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*. pages 820–828.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385* .

Nicolas Heess, Gregory Wayne, David Silver, Tim Lillicrap, Tom Erez, and Yuval Tassa. 2015. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*. pages 2944–2952.

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

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. Lcsts: a large scale chinese short text summarization dataset. *arXiv preprint arXiv:1506.05865* .

Navdeep Jaitly, Quoc V Le, Oriol Vinyals, Ilya Sutskeyver, and Samy Bengio. 2015. An online sequence-to-sequence model using partial conditioning. *arXiv preprint arXiv:1511.04868* .

Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017. Does neural machine translation benefit from larger context? *arXiv preprint arXiv:1704.05135* .

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017a. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734* .

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017b. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics* 5:339–351.

Łukasz Kaiser, Aidan Gomez, and Franccois Chollet. 2017a. Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059* .

Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017b. Learning to remember rare events. *arXiv preprint arXiv:1703.03129* .

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*. pages 1700–1709.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* .

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'16, pages 2741–2749.

Yoon Kim and Alexander Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*.

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

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

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872* .

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 48–54.

Karol Kurach, Marcin Andrychowicz, and Ilya Sutskever. 2015. Neural random-access machines. *arXiv preprint arXiv:1511.06392* .

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science* 350(6266):1332–1338.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *Proceedings of International Conference on Learning Representations (ICLR)*. Vancouver, Canada.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017* .

Jason Lee, Kyunghyun Cho, Jason Weston, and Douwe Kiela. 2017. Emergent translation in multi-agent communication. *arXiv preprint arXiv:1710.06922* .

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016a. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562* .

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017. Learning to decode for future success. *arXiv preprint arXiv:1701.06549* .

Liangyou Li, Andy Way, and Qun Liu. 2016b. Phrase-level combination of smt and tm using constrained word lattice. In *The 54th Annual Meeting of the Association for Computational Linguistics*. page 275.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* .

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*. Association for Computational Linguistics, Barcelona, Spain, pages 74–81.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, page 605.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586* .

Yuping Luo, Chung-Cheng Chiu, Navdeep Jaitly, and Ilya Sutskever. 2016. Learning online alignments with continuous rewards policy gradient. *arXiv preprint arXiv:1608.01281* .

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015a. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114* .

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *arXiv preprint arXiv:1604.00788* .

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015b. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015c. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 11–19.

Randi Martin, Jason Crowther, Meredith Knight, Franklin Tamborello, and Chin-Lung Yang. 2010. Planning in sentence production: Evidence for the phrase as a default planning scope. *Cognition* 116(2):177–192.

Geoffrey J McLachlan and Kaye E Basford. 1988. Mixture models. inference and applications to clustering. *Statistics: Textbooks and Monographs, New York: Dekker, 1988* 1.

Takashi Mieno, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Speed or accuracy? a study in evaluation of simultaneous speech translation. In *INTERSPEECH*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. *INTERSPEECH* 2:3.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126* .

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2017. Meta-learning with temporal convolutions. *arXiv preprint arXiv:1707.03141* .

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030* .

Maria Nadejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Birch. 2017. Syntax-aware neural machine translation using ccg. *arXiv preprint arXiv:1702.01147* .

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. *arXiv preprint arXiv:1704.04572* .

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 551–556.

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

Aaron B Phillips. 2012. *Modeling, Relevance in Statistical Machine Translation: Scoring Aligment, Context, and Annotations of Translation Instances*. Ph.D. thesis, Carnegie Mellon University.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. 2017. Neural episodic control. *arXiv preprint arXiv:1703.01988* .

Vivek Kumar Rangarajan Sridhar, John Chen, Srinivas Bangalore, Andrej Ljolje, and Rathinavelu Chengalvarayan. 2013. Segmentation strategies for streaming speech translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 230–238.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* .

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics* pages 400–407.

David Rumelhart, Geoffrey Hinton, and Ronald Williams. 1986. Learning representations by back-propagating errors. *Nature* pages 323–533.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* .

Harsh Satija and Joelle Pineau. 2016. Simultaneous machine translation using deep reinforcement learning. *Abstraction in Reinforcement Learning Workshop, ICML2016* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015a. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709* .

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. *arXiv preprint arXiv:1606.02891* .

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364* .

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433* .

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *ICML*.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*. pages 4080–4090.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714* .

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387* .

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Dániel Varga. 2006. The jrc-acquis: A multilingual aligned parallel corpus with 20+ languages. *arXiv preprint cs/0609058* .

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*. pages 2431–2439.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *NIPS* .

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4(2).

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2016a. Neural machine translation with reconstruction. *arXiv preprint arXiv:1611.01874* .

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016b. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811* .

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.

Fernanda Viégas, Greg Corrado, Jeffrey Dean, Macduff Hughes, Martin Wattenberg, Maxim Krikun, Melvin Johnson, Mike Schuster, Nikhil Thorat, Quoc V Le, et al. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation .

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*. pages 3630–3638.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In *Advances in Neural Information Processing Systems*. pages 2692–2700.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015b. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2755–2763.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869* .

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017. Exploiting cross-sentence context for neural machine translation. *arXiv preprint arXiv:1704.04347* .

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916* .

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960* .

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *ArXiv e-prints* .

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised neural machine translation with weight sharing. *arXiv preprint arXiv:1804.09057* .

Mahsa Yarmohammadi, Vivek Kumar Rangarajan Sridhar, Srinivas Bangalore, and Baskaran Sankaran. 2013. Incremental segmentation and decoding strategies for simultaneous translation. In *IJCNLP*. pages 1032–1036.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. *arXiv preprint arXiv:1609.08194* .

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1535–1545.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1934–1945.

Ying Zhang and Stephan Vogel. 2005. An efficient phrase-to-phrase alignment model for arbitrarily long phrase and large corpora.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *arXiv preprint arXiv:1601.00710* .

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1568–1575.