A Character-Level Decoder without Explicit Segmentation for Neural Machine Translation

Junyoung Chung
Université de Montréal
junyoung.chung@umontreal.ca

Kyunghyun Cho
New York University

Yoshua Bengio
Université de Montréal
CIFAR Senior Fellow

Abstract

The existing machine translation systems, whether phrase-based or neural, have relied almost exclusively on word-level modelling with explicit segmentation. In this paper, we ask a fundamental question: can neural machine translation generate a character sequence without any explicit segmentation? To answer this question, we evaluate an attention-based encoder–decoder with a subword-level encoder and a character-level decoder on four language pairs–En-Cs, En-De, En-Ru and En-Fi– using the parallel corpora from WMT'15. Our experiments show that the models with a character-level decoder outperform the ones with a subword-level decoder on all of the four language pairs. Furthermore, the ensembles of neural models with a character-level decoder outperform the state-of-the-art non-neural machine translation systems on En-Cs, En-De and En-Fi and perform comparably on En-Ru.

1 Introduction

The existing machine translation systems have relied almost exclusively on word-level modelling with explicit segmentation. This is mainly due to the issue of data sparsity which becomes much more severe, especially for n-grams, when a sentence is represented as a sequence of characters rather than words, as the length of the sequence grows significantly. In addition to data sparsity, we often have a priori belief that a word, or its segmented-out lexeme, is a basic unit of meaning, making it natural to approach translation as mapping from a sequence of source-language words to a sequence of target-language words.

This has continued with the more recently proposed paradigm of neural machine translation, although neural networks do not suffer from character-level modelling and rather suffer from the issues specific to word-level modelling, such as the increased computational complexity from a very large target vocabulary (Jean et al., 2015; Luong et al., 2015b). Therefore, in this paper, we address a question of whether *neural machine translation can be done directly on a sequence of characters without any explicit word segmentation*.

To answer this question, we focus on representing the target side as a character sequence. We evaluate neural machine translation models with a character-level decoder on four language pairs from WMT'15 to make our evaluation as convincing as possible. We represent the source side as a sequence of subwords extracted using byte-pair encoding from Sennrich et al. (2015), and vary the target side to be either a sequence of subwords or characters. On the target side, we further design a novel recurrent neural network (RNN), called *bi-scale recurrent network*, that better handles multiple timescales in a sequence, and test it in addition to a naive, stacked recurrent neural network.

On all of the four language pairs–En-Cs, En-De, En-Ru and En-Fi–, the models with a character-level decoder outperformed the ones with a subword-level decoder. We observed a similar trend with the ensemble of each of these configurations, outperforming both the previous best neural and non-neural translation systems on En-Cs, En-De and En-Fi, while achieving a comparable result on En-Ru. We find these results to be a strong evidence that neural machine translation can indeed learn to translate at the character-level and that in fact, it benefits from doing so.

2 Neural Machine Translation

Neural machine translation refers to a recently proposed approach to machine translation (Cho et

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1693–1703,
Berlin, Germany, August 7-12, 2016. ©2016 Association for Computational Linguistics

al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015). This approach aims at building an end-to-end neural network that takes as input a source sentence $X = (x_1, \ldots, x_{T_x})$ and outputs its translation $Y = (y_1, \ldots, y_{T_y})$, where x_t and $y_{t'}$ are respectively source and target symbols. This neural network is constructed as a composite of an encoder network and a decoder network.

The encoder network encodes the input sentence X into its continuous representation. In this paper, we closely follow the neural translation model proposed in Bahdanau et al. (2015) and use a bidirectional recurrent neural network, which consists of two recurrent neural networks. The forward network reads the input sentence in a forward direction: $\overrightarrow{\mathbf{z}}_t = \overrightarrow{\phi}(e_x(x_t), \overrightarrow{\mathbf{z}}_{t-1})$, where $e_x(x_t)$ is a continuous embedding of the t-th input symbol, and ϕ is a recurrent activation function. Similarly, the reverse network reads the sentence in a reverse direction (right to left): $\overleftarrow{\mathbf{z}}_t = \overleftarrow{\phi}(e_x(x_t), \overleftarrow{\mathbf{z}}_{t+1})$. At each location in the input sentence, we concatenate the hidden states from the forward and reverse RNNs to form a context set $C = \{\mathbf{z}_1, \ldots, \mathbf{z}_{T_x}\}$, where $\mathbf{z}_t = \left[\overrightarrow{\mathbf{z}}_t; \overleftarrow{\mathbf{z}}_t\right]$.

Then the decoder computes the conditional distribution over all possible translations based on this context set. This is done by first rewriting the conditional probability of a translation: $\log p(Y|X) = \sum_{t'=1}^{T_y} \log p(y_{t'}|y_{<t'}, X)$. For each conditional term in the summation, the decoder RNN updates its hidden state by

$$\mathbf{h}_{t'} = \phi(e_y(y_{t'-1}), \mathbf{h}_{t'-1}, \mathbf{c}_{t'}), \quad (1)$$

where e_y is the continuous embedding of a target symbol. $\mathbf{c}_{t'}$ is a context vector computed by a soft-alignment mechanism:

$$\mathbf{c}_{t'} = f_{\text{align}}(e_y(y_{t'-1}), \mathbf{h}_{t'-1}, C)). \quad (2)$$

The soft-alignment mechanism f_{align} weights each vector in the context set C according to its relevance given what has been translated. The weight of each vector $\mathbf{z}_t$ is computed by

$$\alpha_{t,t'} = \frac{1}{Z} e^{f_{\text{score}}(e_y(y_{t'-1}), \mathbf{h}_{t'-1}, \mathbf{z}_t)}, \quad (3)$$

where f_{score} is a parametric function returning an unnormalized score for $\mathbf{z}_t$ given $\mathbf{h}_{t'-1}$ and $y_{t'-1}$. We use a feedforward network with a single hidden layer in this paper.[1] Z is a normalization constant: $Z = \sum_{k=1}^{T_x} e^{f_{\text{score}}(e_y(y_{t'-1}), \mathbf{h}_{t'-1}, \mathbf{z}_k)}$. This procedure can be understood as computing the alignment probability between the t'-th target symbol and t-th source symbol.

The hidden state $\mathbf{h}_{t'}$, together with the previous target symbol $y_{t'-1}$ and the context vector $\mathbf{c}_{t'}$, is fed into a feedforward neural network to result in the conditional distribution:

$$p(y_{t'} \mid y_{<t'}, X) \propto e^{f_{\text{out}}^{y_{t'}}(e_y(y_{t'-1}), \mathbf{h}_{t'}, \mathbf{c}_{t'})}. \quad (4)$$

The whole model, consisting of the encoder, decoder and soft-alignment mechanism, is then tuned end-to-end to minimize the negative log-likelihood using stochastic gradient descent.

3 Towards Character-Level Translation

3.1 Motivation

Let us revisit how the source and target sentences (X and Y) are represented in neural machine translation. For the source side of any given training corpus, we scan through the whole corpus to build a vocabulary V_x of unique tokens to which we assign integer indices. A source sentence X is then built as a sequence of the indices of such tokens belonging to the sentence, i.e., $X = (x_1, \ldots, x_{T_x})$, where $x_t \in \{1, 2, \ldots, |V_x|\}$. The target sentence is similarly transformed into a target sequence of integer indices.

Each token, or its index, is then transformed into a so-called one-hot vector of dimensionality $|V_x|$. All but one elements of this vector are set to 0. The only element whose index corresponds to the token's index is set to 1. This one-hot vector is the one which any neural machine translation model sees. The embedding function, e_x or e_y, is simply the result of applying a linear transformation (the embedding matrix) to this one-hot vector.

The important property of this approach based on one-hot vectors is that the neural network is oblivious to the underlying semantics of the tokens. To the neural network, each and every token in the vocabulary is equal distance away from every other token. The semantics of those tokens are simply *learned* (into the embeddings) to maximize the translation quality, or the log-likelihood of the model.

This property allows us great freedom in the choice of tokens' unit. Neural networks have been shown to work well with word tokens (Bengio et al., 2001; Schwenk, 2007; Mikolov et al., 2010) but also with finer units, such as subwords (Sennrich et al., 2015; Botha and Blunsom, 2014; Lu-

[1] For other possible implementations, see (Luong et al., 2015a).

ong et al., 2013) as well as symbols resulting from compression/encoding (Chitnis and DeNero, 2015). Although there have been a number of previous research reporting the use of neural networks with characters (see, e.g., Mikolov et al. (2012) and Santos and Zadrozny (2014)), the dominant approach has been to preprocess the text into a sequence of symbols, each associated with a sequence of characters, after which the neural network is presented with those symbols rather than with characters.

More recently in the context of neural machine translation, two research groups have proposed to directly use characters. Kim et al. (2015) proposed to represent each word not as a single integer index as before, but as a sequence of characters, and use a convolutional network followed by a highway network (Srivastava et al., 2015) to extract a continuous representation of the word. This approach, which effectively replaces the embedding function e_x, was adopted by Costa-Jussà and Fonollosa (2016) for neural machine translation. Similarly, Ling et al. (2015b) use a bidirectional recurrent neural network to replace the embedding functions e_x and e_y to respectively encode a character sequence to and from the corresponding continuous word representation. A similar, but slightly different approach was proposed by Lee et al. (2015), where they explicitly mark each character with its relative location in a word (e.g., "B"eginning and "I"ntermediate).

Despite the fact that these recent approaches work at the level of characters, it is less satisfying that they all rely on knowing how to segment characters into words. Although it is generally easy for languages like English, this is not always the case. This word segmentation procedure can be as simple as tokenization followed by some punctuation normalization, but also can be as complicated as morpheme segmentation requiring a separate model to be trained in advance (Creutz and Lagus, 2005; Huang and Zhao, 2007). Furthermore, these segmentation[2] steps are often tuned or designed separately from the ultimate objective of translation quality, potentially contributing to a suboptimal quality.

Based on this observation and analysis, in this paper, we ask ourselves and the readers a question which should have been asked much earlier: *Is it possible to do character-level translation without any explicit segmentation?*

3.2 Why Word-Level Translation?

(1) Word as a Basic Unit of Meaning A word can be understood in two different senses. In the abstract sense, a word is a basic unit of meaning (lexeme), and in the other sense, can be understood as a "concrete word as used in a sentence." (Booij, 2012). A word in the former sense turns into that in the latter sense via a process of morphology, including inflection, compounding and derivation. These three processes do alter the meaning of the lexeme, but often it stays close to the original meaning. Because of this view of words as basic units of meaning (either in the form of lexemes or derived form) from linguistics, much of previous work in natural language processing has focused on using words as basic units of which a sentence is encoded as a sequence. Also, the potential difficulty in finding a mapping between a word's character sequence and meaning[3] has likely contributed to this trend toward word-level modelling.

(2) Data Sparsity There is a further technical reason why much of previous research on machine translation has considered words as a basic unit. This is mainly due to the fact that major components in the existing translation systems, such as language models and phrase tables, are a count-based estimator of probabilities. In other words, a probability of a subsequence of symbols, or pairs of symbols, is estimated by counting the number of its occurrences in a training corpus. This approach severely suffers from the issue of data sparsity, which is due to a large state space which grows exponentially w.r.t. the length of subsequences while growing only linearly w.r.t. the corpus size. This poses a great challenge to character-level modelling, as any subsequence will be on average 4–5 times longer when characters, instead of words, are used. Indeed, Vilar et al. (2007) reported worse performance when the character sequence was directly used by a phrase-based machine translation system. More recently, Neubig et al. (2013) proposed a method to improve character-level translation with phrase-based translation systems, however, with only a limited success.

[2] From here on, the term *segmentation* broadly refers to any method that splits a given character sequence into a sequence of subword symbols.

[3] For instance, "quit", "quite" and "quiet" are one edit-distance away from each other but have distinct meanings.

(3) Vanishing Gradient Specifically to neural machine translation, a major reason behind the wide adoption of word-level modelling is due to the difficulty in modelling long-term dependencies with recurrent neural networks (Bengio et al., 1994; Hochreiter, 1998). As the lengths of the sentences on both sides grow when they are represented in characters, it is easy to believe that there will be more long-term dependencies that must be captured by the recurrent neural network for successful translation.

3.3 Why Character-Level Translation?

Why *not* Word-Level Translation? The most pressing issue with word-level processing is that we do not have a perfect word segmentation algorithm for any one language. A perfect segmentation algorithm needs to be able to segment any given sentence into a sequence of lexemes and morphemes. This problem is however a difficult problem on its own and often requires decades of research (see, e.g., Creutz and Lagus (2005) for Finnish and other morphologically rich languages and Huang and Zhao (2007) for Chinese). Therefore, many opt to using either a rule-based tokenization approach or a suboptimal, but still available, learning based segmentation algorithm.

The outcome of this naive, sub-optimal segmentation is that the vocabulary is often filled with many similar words that share a lexeme but have different morphology. For instance, if we apply a simple tokenization script to an English corpus, "run", "runs", "ran" and "running" are all separate entries in the vocabulary, while they clearly share the same lexeme "run". This prevents any machine translation system, in particular neural machine translation, from modelling these morphological variants efficiently.

More specifically in the case of neural machine translation, each of these morphological variants– "run", "runs", "ran" and "running"– will be assigned a d-dimensional word vector, leading to four independent vectors, while it is clear that if we can segment those variants into a lexeme and other morphemes, we can model them more efficiently. For instance, we can have a d-dimensional vector for the lexeme "run" and much smaller vectors for "s" and"ing". Each of those variants will be then a composite of the lexeme vector (shared across these variants) and morpheme vectors (shared across words sharing the same suffix, for example) (Botha and Blunsom, 2014). This makes use of distributed representation, which generally yields better generalization, but seems to require an optimal segmentation, which is unfortunately almost never available.

In addition to inefficiency in modelling, there are two additional negative consequences from using (unsegmented) words. First, the translation system cannot generalize well to novel words, which are often mapped to a token reserved for an unknown word. This effectively ignores any meaning or structure of the word to be incorporated when translating. Second, even when a lexeme is common and frequently observed in the training corpus, its morphological variant may not be. This implies that the model sees this specific, rare morphological variant much less and will not be able to translate it well. However, if this rare morphological variant shares a large part of its spelling with other more common words, it is desirable for a machine translation system to exploit those common words when translating those rare variants.

Why Character-Level Translation? All of these issues can be addressed to certain extent by directly modelling characters. Although the issue of data sparsity arises in character-level translation, it is elegantly addressed by using a parametric approach based on recurrent neural networks instead of a non-parametric count-based approach. Furthermore, in recent years, we have learned how to build and train a recurrent neural network that can well capture long-term dependencies by using more sophisticated activation functions, such as long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997) and gated recurrent units (Cho et al., 2014).

Kim et al. (2015) and Ling et al. (2015a) recently showed that by having a neural network that converts a character sequence into a word vector, we avoid the issues from having many morphological variants appearing as separate entities in a vocabulary. This is made possible by sharing the character-to-word neural network across all the unique tokens. A similar approach was applied to machine translation by Ling et al. (2015b).

These recent approaches, however, still rely on the availability of a good, if not optimal, segmentation algorithm. Ling et al. (2015b) indeed states that "[m]uch of the prior information regarding morphology, cognates and rare word translation among others, should be incorporated”.

It however becomes unnecessary to consider these prior information, if we use a neural network, be it recurrent, convolution or their combination, directly on the unsegmented character sequence. The possibility of using a sequence of unsegmented characters has been studied over many years in the field of deep learning. For instance, Mikolov et al. (2012) and Sutskever et al. (2011) trained a recurrent neural network language model (RNN-LM) on character sequences. The latter showed that it is possible to generate sensible text sequences by simply sampling a character at a time from this model. More recently, Zhang et al. (2015) and Xiao and Cho (2016) successfully applied a convolutional net and a convolutional-recurrent net respectively to character-level document classification without any explicit segmentation. Gillick et al. (2015) further showed that it is possible to train a recurrent neural network on unicode bytes, instead of characters or words, to perform part-of-speech tagging and named entity recognition.

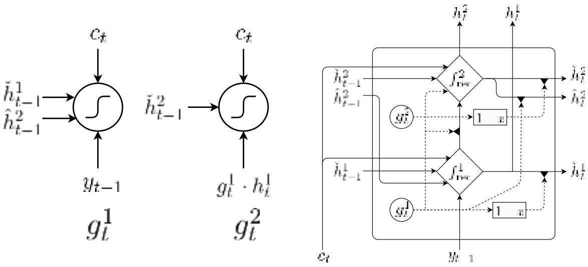
These previous works suggest the possibility of applying neural networks for the task of machine translation, which is often considered a substantially more difficult problem compared to document classification and language modelling.

3.4 Challenges and Questions

There are two overlapping sets of challenges for the source and target sides. On the source side, it is unclear how to build a neural network that learns a highly nonlinear mapping from a spelling to the meaning of a sentence.

On the target side, there are two challenges. The first challenge is the same one from the source side, as the decoder neural network needs to summarize what has been translated. In addition to this, the character-level modelling on the target side is more challenging, as the decoder network must be able to generate a long, coherent sequence of characters. This is a great challenge, as the size of the state space grows exponentially w.r.t. the number of symbols, and in the case of characters, it is often 300-1000 symbols long.

All these challenges should first be framed as questions; whether the current recurrent neural networks, which are already widely used in neural machine translation, are able to address these challenges as they are. In this paper, we aim at answering these *questions* empirically and focus on the challenges on the target side (as the target side shows both of the challenges).



(a) Gating units (b) One-step processing

Figure 1: Bi-scale recurrent neural network

4 Character-Level Translation

In this paper, we try to answer the questions posed earlier by testing two different types of recurrent neural networks on the target side (decoder).

First, we test an existing recurrent neural network with gated recurrent units (GRUs). We call this decoder a *base* decoder.

Second, we build a novel two-layer recurrent neural network, inspired by the gated-feedback network from Chung et al. (2015), called a *bi-scale* recurrent neural network. We design this network to facilitate capturing two timescales, motivated by the fact that characters and words may work at two separate timescales.

We choose to test these two alternatives for the following purposes. Experiments with the base decoder will clearly answer whether the existing neural network is enough to handle character-level decoding, which has not been properly answered in the context of machine translation. The alternative, the bi-scale decoder, is tested in order to see whether it is possible to design a better decoder, if the answer to the first question is positive.

4.1 Bi-Scale Recurrent Neural Network

In this proposed bi-scale recurrent neural network, there are two sets of hidden units, $\mathbf{h}^1$ and $\mathbf{h}^2$. They contain the same number of units, i.e., $\text{dim}(\mathbf{h}^1) = \text{dim}(\mathbf{h}^2)$. The first set $\mathbf{h}^1$ models a fast-changing timescale (thereby, a *faster layer*), and $\mathbf{h}^2$ a slower timescale (thereby, a *slower layer*). For each hidden unit, there is an associated gating unit, to which we refer by $\mathbf{g}^1$ and $\mathbf{g}^2$. For the description below, we use $y_{t'-1}$ and $\mathbf{c}_{t'}$ for the previous target symbol and the context vector (see Eq. (2)), respectively.

Let us start with the faster layer. The faster layer outputs two sets of activations, a normal output $\mathbf{h}^1_{t'}$ and its gated version $\check{\mathbf{h}}^1_{t'}$. The activation of the faster layer is computed by

$$\mathbf{h}^1_{t'} = \tanh\left(\mathbf{W}^{h^1}\left[e_y(y_{t'-1}); \check{\mathbf{h}}^1_{t'-1}; \hat{\mathbf{h}}^2_{t'-1}; \mathbf{c}_{t'}\right]\right),$$

where $\check{\mathbf{h}}^1_{t'-1}$ and $\hat{\mathbf{h}}^2_{t'-1}$ are the gated activations of the faster and slower layers respectively. These gated activations are computed by

$$\check{\mathbf{h}}^1_{t'} = (1 - \mathbf{g}^1_{t'}) \odot \mathbf{h}^1_{t'}, \;\; \hat{\mathbf{h}}^2_{t'} = \mathbf{g}^1_{t'} \odot \mathbf{h}^2_{t'}.$$

In other words, the faster layer's activation is based on the adaptive combination of the faster and slower layers' activations from the previous time step. Whenever the faster layer determines that it needs to reset, i.e., $\mathbf{g}^1_{t'-1} \approx 1$, the next activation will be determined based more on the slower layer's activation.

The faster layer's gating unit is computed by

$$\mathbf{g}^1_{t'} = \sigma\left(\mathbf{W}^{g^1}\left[e_y(y_{t'-1}); \check{\mathbf{h}}^1_{t'-1}; \hat{\mathbf{h}}^2_{t'-1}; \mathbf{c}_{t'}\right]\right),$$

where σ is a sigmoid function.

The slower layer also outputs two sets of activations, a normal output $\mathbf{h}^2_{t'}$ and its gated version $\check{\mathbf{h}}^2_{t'}$. These activations are computed as follows:

$$\begin{aligned}\mathbf{h}^2_{t'} &= (1 - \mathbf{g}^1_{t'}) \odot \mathbf{h}^2_{t'-1} + \mathbf{g}^1_{t'} \odot \tilde{\mathbf{h}}^2_{t'},\\ \check{\mathbf{h}}^2_{t'} &= (1 - \mathbf{g}^2_{t'}) \odot \mathbf{h}^2_{t'},\end{aligned}$$

where $\tilde{\mathbf{h}}^2_{t'}$ is a candidate activation. The slower layer's gating unit $\mathbf{g}^2_{t'}$ is computed by

$$\mathbf{g}^2_{t'} = \sigma\left(\mathbf{W}^{g^2}\left[(\mathbf{g}^1_{t'} \odot \mathbf{h}^1_{t'}); \check{\mathbf{h}}^2_{t'-1}; \mathbf{c}_{t'}\right]\right).$$

This adaptive leaky integration based on the gating unit from the faster layer has a consequence that the slower layer updates its activation only when the faster layer resets. This puts a soft constraint that the faster layer runs at a faster rate by preventing the slower layer from updating while the faster layer is processing a current chunk.

The candidate activation is then computed by

$$\tilde{\mathbf{h}}^2_{t'} = \tanh\left(\mathbf{W}^{h^2}\left[(\mathbf{g}^1_{t'} \odot \mathbf{h}^1_{t'}); \check{\mathbf{h}}^2_{t'-1}; \mathbf{c}_{t'}\right]\right). \quad (5)$$

$\check{\mathbf{h}}^2_{t'-1}$ indicates the *reset* activation from the previous time step, similarly to what happened in the faster layer, and $\mathbf{c}_{t'}$ is the input from the context.

According to $\mathbf{g}^1_{t'} \odot \mathbf{h}^1_{t'}$ in Eq. (5), the faster layer influences the slower layer, only when the faster layer has finished processing the current chunk and is about to *reset* itself ($\mathbf{g}^1_{t'} \approx 1$). In other words, the slower layer does not receive any input from the faster layer, until the faster layer has quickly processed the current chunk, thereby running at a slower rate than the faster layer does.

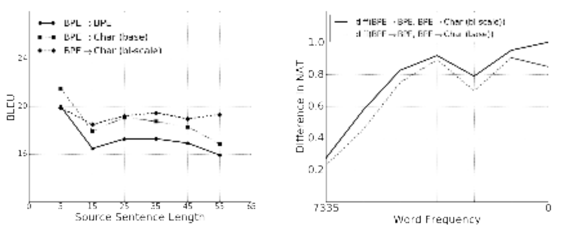


Figure 2: (left) The BLEU scores on En-Cs w.r.t. the length of source sentences. (right) The difference of word negative log-probabilities between the subword-level decoder and either of the character-level base or bi-scale decoder.

At each time step, the final output of the proposed bi-scale recurrent neural network is the concatenation of the output vectors of the faster and slower layers, i.e., $\left[\mathbf{h}^1; \mathbf{h}^2\right]$. This concatenated vector is used to compute the probability distribution over all the symbols in the vocabulary, as in Eq. (4). See Fig. 1 for graphical illustration.

5 Experiment Settings

For evaluation, we represent a source sentence as a sequence of subword symbols extracted by byte-pair encoding (BPE, Sennrich et al. (2015)) and a target sentence either as a sequence of BPE-based symbols or as a sequence of characters.

Corpora and Preprocessing We use all available parallel corpora for four language pairs from WMT'15: En-Cs, En-De, En-Ru and En-Fi. They consist of 12.1M, 4.5M, 2.3M and 2M sentence pairs, respectively. We tokenize each corpus using a tokenization script included in Moses.[4] We only use the sentence pairs, when the source side is up to 50 subword symbols long and the target side is either up to 100 subword symbols or 500 characters. We do not use any monolingual corpus.

For all the pairs other than En-Fi, we use newstest-2013 as a development set, and newstest-2014 (Test$_1$) and newstest-2015 (Test$_2$) as test sets. For En-Fi, we use newsdev-2015 and newstest-2015 as development and test sets, respectively.

[4] Although tokenization is not necessary for character-level modelling, we tokenize the all target side corpora to make comparison against word-level modelling easier.

		Src	Trgt	Depth	Attention		Model	Development		Test$_1$		Test$_2$	
					$\mathbf{h}^1$	$\mathbf{h}^2$		Single	Ens	Single	Ens	Single	Ens
En-De	(a)	BPE	BPE	1	✓		Base	20.78	–	19.98	–	21.72	–
	(b)			2	✓	✓		$21.26^{21.45}_{20.62}$	23.49	$20.47^{20.88}_{19.30}$	23.10	$22.02^{22.21}_{21.35}$	24.83
	(c)		Char	2		✓	Base	$21.57^{21.88}_{20.88}$	23.14	$\mathbf{21.33}^{21.56}_{19.82}$	$\underline{23.11}$	$\mathbf{23.45}^{23.91}_{21.72}$	25.24
	(d)			2	✓	✓		20.31	–	19.70	–	21.30	–
	(e)			2		✓	Bi-S	$21.29^{21.43}_{21.13}$	23.05	$21.25^{21.47}_{20.62}$	23.04	$23.06^{23.47}_{22.85}$	$\underline{25.44}$
	(f)			2	✓	✓		20.78	–	20.19	–	22.26	–
	(g)			2	✓			20.08	–	19.39	–	20.94	–
	State-of-the-art Non-Neural Approach*							–		$20.60^{(1)}$		$24.00^{(2)}$	
En-Cs	(h)	BPE	BPE	2	✓	✓	Base	$16.12^{16.96}_{15.96}$	19.21	$17.16^{17.68}_{16.38}$	20.79	$14.63^{15.09}_{14.26}$	17.61
	(i)		Char	2		✓	Base	$17.68^{17.78}_{17.39}$	19.52	$19.25^{19.55}_{18.89}$	21.95	$\mathbf{16.98}^{17.17}_{16.81}$	18.92
	(j)			2		✓	Bi-S	$17.62^{17.93}_{17.43}$	19.83	$\mathbf{19.27}^{19.53}_{19.15}$	$\underline{22.15}$	$16.86^{17.10}_{16.68}$	$\underline{18.93}$
	State-of-the-art Non-Neural Approach*							–		$21.00^{(3)}$		$18.20^{(4)}$	
En-Ru	(k)	BPE	BPE	2	✓	✓	Base	$18.56^{18.70}_{18.26}$	21.17	$25.30^{25.40}_{24.95}$	29.26	$19.72^{20.29}_{19.02}$	22.96
	(l)		Char	2		✓	Base	$18.56^{18.87}_{18.39}$	20.53	$\mathbf{26.00}^{26.07}_{25.04}$	$\underline{29.37}$	$\mathbf{21.10}^{21.24}_{20.14}$	23.51
	(m)			2		✓	Bi-S	$18.30^{18.54}_{17.88}$	20.53	$25.59^{25.76}_{24.57}$	29.26	$20.73^{21.02}_{19.97}$	$\underline{23.75}$
	State-of-the-art Non-Neural Approach*							–		$28.70^{(5)}$		$24.30^{(6)}$	
En-Fi	(n)	BPE	BPE	2	✓	✓	Base	$9.61^{10.02}_{9.24}$	11.92	–	–	$8.97^{9.17}_{8.88}$	11.73
	(o)		Char	2		✓	Base	$11.19^{11.55}_{11.09}$	13.72	–	–	$\mathbf{10.93}^{11.56}_{10.11}$	$\underline{13.48}$
	(p)			2		✓	Bi-S	$10.73^{11.04}_{10.40}$	13.39	–	–	$10.24^{10.63}_{9.71}$	13.32
	State-of-the-art Non-Neural Approach*							–		–		$12.70^{(7)}$	

Table 1: BLEU scores of the subword-level, character-level base and character-level bi-scale decoders for both single models and ensembles. The best scores among the single models per language pair are bold-faced, and those among the ensembles are underlined. When available, we report the median value, and the minimum and maximum values as a subscript and a superscript, respectively. (∗) `http://matrix.statmt.org/` as of 11 March 2016 (constrained only). (1) Freitag et al. (2014). (2, 6) Williams et al. (2015). (3, 5) Durrani et al. (2014). (4) Haddow et al. (2015). (7) Rubino et al. (2015).

Models and Training We test three models settings: (1) BPE→BPE, (2) BPE→Char (base) and (3) BPE→Char (bi-scale). The latter two differ by the type of recurrent neural network we use. We use GRUs for the encoder in all the settings. We used GRUs for the decoders in the first two settings, (1) and (2), while the proposed bi-scale recurrent network was used in the last setting, (3). The encoder has 512 hidden units for each direction (forward and reverse), and the decoder has 1024 hidden units per layer.

We train each model using stochastic gradient descent with Adam (Kingma and Ba, 2014). Each update is computed using a minibatch of 128 sentence pairs. The norm of the gradient is clipped with a threshold 1 (Pascanu et al., 2013).

Decoding and Evaluation We use beamsearch to approximately find the most likely translation given a source sentence. The beam widths are 5 and 15 respectively for the subword-level and character-level decoders. They were chosen based on the translation quality on the development set. The translations are evaluated using BLEU.[5]

Multilayer Decoder and Soft-Alignment Mechanism When the decoder is a multilayer recurrent neural network (including a stacked network as well as the proposed bi-scale network), the decoder outputs multiple hidden vectors–$\left\{\mathbf{h}^1, \ldots, \mathbf{h}^L\right\}$ for L layers, at a time. This allows an extra degree of freedom in the soft-alignment mechanism (f_{score} in Eq. (3)). We evaluate using alternatives, including (1) using only $\mathbf{h}^L$ (slower layer) and (2) using all of them (concatenated).

Ensembles We also evaluate an ensemble of neural machine translation models and compare its performance against the state-of-the-art phrase-based translation systems on all four language pairs. We decode from an ensemble by taking the average of the output probabilities at each step.

6 Quantitative Analysis

Slower Layer for Alignment On En-De, we test which layer of the decoder should be used

[5] We used the multi-bleu.perl script from Moses.

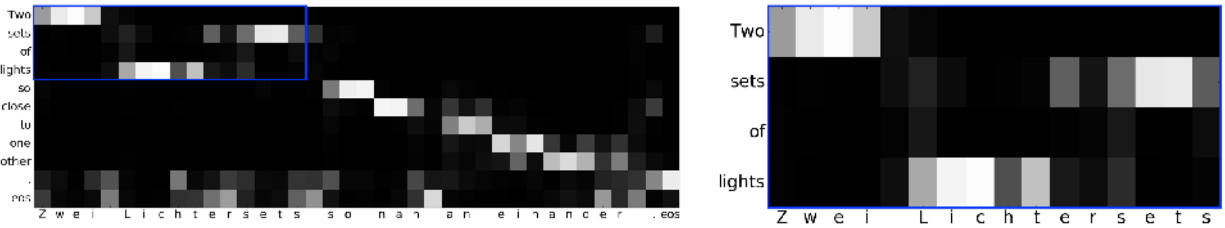


Figure 3: Alignment matrix of a test example from En-De using the BPE→Char (bi-scale) model.

for computing soft-alignments. In the case of subword-level decoder, we observed no difference between choosing any of the two layers of the decoder against using the concatenation of all the layers (Table 1 (a–b)) On the other hand, with the character-level decoder, we noticed an improvement when only the slower layer ($\mathbf{h}^2$) was used for the soft-alignment mechanism (Table 1 (c–g)). This suggests that the soft-alignment mechanism benefits by aligning a larger chunk in the target with a subword unit in the source, and we use only the slower layer for all the other language pairs.

Single Models In Table 1, we present a comprehensive report of the translation qualities of (1) subword-level decoder, (2) character-level base decoder and (3) character-level bi-scale decoder, for all the language pairs. We see that the both types of character-level decoder outperform the subword-level decoder for En-Cs and En-Fi quite significantly. On En-De, the character-level base decoder outperforms both the subword-level decoder and the character-level bi-scale decoder, validating the effectiveness of the character-level modelling. On En-Ru, among the single models, the character-level decoders outperform the subword-level decoder, but in general, we observe that all the three alternatives work comparable to each other.

These results clearly suggest that *it is indeed possible to do character-level translation without explicit segmentation.* In fact, what we observed is that character-level translation often surpasses the translation quality of word-level translation. Of course, we note once again that our experiment is restricted to using an unsegmented character sequence at the decoder only, and a further exploration toward replacing the source sentence with an unsegmented character sequence is needed.

Ensembles Each ensemble was built using eight independent models. The first observation we make is that in all the language pairs, neural machine translation performs comparably to, or often better than, the state-of-the-art non-neural translation system. Furthermore, the character-level decoders outperform the subword-level decoder in all the cases.

7 Qualitative Analysis

(1) Can the character-level decoder generate a long, coherent sentence? The translation in characters is dramatically longer than that in words, likely making it more difficult for a recurrent neural network to generate a coherent sentence in characters. This belief turned out to be false. As shown in Fig. 2 (left), there is no significant difference between the subword-level and character-level decoders, even though the lengths of the generated translations are generally 5–10 times longer in characters.

(2) Does the character-level decoder help with rare words? One advantage of character-level modelling is that it can model the composition of any character sequence, thereby better modelling rare morphological variants. We empirically confirm this by observing the growing gap in the average negative log-probability of words between the subword-level and character-level decoders as the frequency of the words decreases. This is shown in Fig. 2 (right) and explains one potential cause behind the success of character-level decoding in our experiments (we define $\text{diff}(x, y) = x - y$).

(3) Can the character-level decoder soft-align between a source word and a target character? In Fig. 3 (left), we show an example soft-alignment of a source sentence, "Two sets of light so close to one another". It is clear that the character-level translation model well captured the alignment between the source subwords and target characters. We observe that the character-level decoder correctly aligns to "lights" and "sets of" when generating a German compound word

"Lichtersets" (see Fig. 3 (right) for the zoomed-in version). This type of behaviour happens similarly between "one another" and "einander". Of course, this does not mean that there exists an alignment between a source word and a target character. Rather, this suggests that the internal state of the character-level decoder, the base or bi-scale, well captures the meaningful chunk of characters, allowing the model to map it to a larger chunk (subword) in the source.

(4) How fast is the decoding speed of the character-level decoder? We evaluate the decoding speed of subword-level base, character-level base and character-level bi-scale decoders on newstest2013 corpus (En-De) with a single Titan X GPU. The subword-level base decoder generates 31.9 words per second, and the character-level base decoder and character-level bi-scale decoder generate 27.5 words per second and 25.6 words per second, respectively. Note that this is evaluated in an online setting, performing consecutive translation, where only one sentence is translated at a time. Translating in a batch setting could differ from these results.

8 Conclusion

In this paper, we addressed a fundamental question on whether a recently proposed neural machine translation system can directly handle translation at the level of characters without any word segmentation. We focused on the target side, in which a decoder was asked to generate one character at a time, while soft-aligning between a target character and a source subword. Our extensive experiments, on four language pairs–En-Cs, En-De, En-Ru and En-Fi– strongly suggest that it is indeed possible for neural machine translation to translate at the level of characters, and that it actually benefits from doing so.

Our result has one limitation that we used subword symbols in the source side. However, this has allowed us a more fine-grained analysis, but in the future, a setting where the source side is also represented as a character sequence must be investigated.

Acknowledgments

The authors would like to thank the developers of Theano (Team et al., 2016). We acknowledge the support of the following agencies for research funding and computing support: NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs, CIFAR and Samsung. KC thanks the support by Facebook, Google (Google Faculty Award 2016) and NVIDIA (GPU Center of Excellence 2015-2016). JC thanks Orhan Firat for his constructive feedbacks.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2001. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, pages 932–938.

Geert Booij. 2012. *The grammar of words: An introduction to linguistic morphology*. Oxford University Press.

Jan A Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *ICML 2014*.

Rohan Chitnis and John DeNero. 2015. Variable-length word encodings for neural translation models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2088–2093.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, October.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*.

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

Mathias Creutz and Krista Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology.

Nadir Durrani, Barry Haddow, Philipp Koehn, and Kenneth Heafield. 2014. Edinburgh's phrase-based machine translation systems for wmt-14. In *Proceedings of the ACL 2014 Ninth Workshop on Statistical Machine Translation, Baltimore, MD, USA*, pages 97–104.

Markus Freitag, Stephan Peitz, Joern Wuebker, Hermann Ney, Matthias Huck, Rico Sennrich, Nadir Durrani, Maria Nadejde, Philip Williams, Philipp Koehn, et al. 2014. Eu-bridge mt: Combined machine translation.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2015. Multilingual language processing from bytes. *arXiv preprint arXiv:1512.00103*.

Barry Haddow, Matthias Huck, Alexandra Birch, Nikolay Bogoychev, and Philipp Koehn. 2015. The edinburgh/jhu phrase-based machine translation systems for wmt 2015. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 126–133.

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

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Changning Huang and Hai Zhao. 2007. Chinese word segmentation: A decade review. *Journal of Chinese Information Processing*, 21(3):8–20.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

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

Hyoung-Gyu Lee, JaeSong Lee, Jun-Seok Kim, and Chang-Ki Lee. 2015. Naver machine translation system for wat 2015. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 69–73.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015a. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

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

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113.

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

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

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

Tomas Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and J Cernocky. 2012. Subword language modeling with neural networks. *Preprint*.

Graham Neubig, Taro Watanabe, Shinsuke Mori, and Tatsuya Kawahara. 2013. Substring-based machine translation. *Machine translation*, 27(2):139–166.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Raphael Rubino, Tommi Pirinen, Miquel Espla-Gomis, N Ljubešic, Sergio Ortiz Rojas, Vassilis Papavassiliou, Prokopis Prokopidis, and Antonio Toral. 2015. Abu-matran at wmt 2015 translation task: Morphological segmentation and web crawling. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 184–191.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

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

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in Neural Information Processing Systems*, pages 2368–2376.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*, pages 1017–1024.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. 2016. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*.

David Vilar, Jan-T Peter, and Hermann Ney. 2007. Can we translate letters? In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 33–39. Association for Computational Linguistics.

Philip Williams, Rico Sennrich, Maria Nadejde, Matthias Huck, and Philipp Koehn. 2015. Edinburgh's syntax-based systems at wmt 2015. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 199–209.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.