NYU-MILA Neural Machine Translation Systems for WMT'16

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

Kyunghyun Cho
New York University

Yoshua Bengio
Université de Montréal
CIFAR Senior Fellow

Abstract

We describe the neural machine translation system of New York University (NYU) and University of Montreal (MILA) for the translation tasks of WMT'16. The main goal of NYU-MILA submission to WMT'16 is to evaluate a new character-level decoding approach in neural machine translation on various language pairs. The proposed neural machine translation system is an attention-based encoder–decoder with a subword-level encoder and a character-level decoder. The decoder of the neural machine translation system does not require explicit segmentation, when characters are used as tokens. The character-level decoding approach provides benefits especially when translating a source language into other morphologically rich languages.

1 Introduction

Word-level modelling with explicit segmentation has been a standard approach in statistical machine translation systems. This is mainly due to the issue of data sparsity, caused by the, exponential growth of the state space as the length of sequences grows larger. This becomes much more severe when a sequence is represented with characters. In addition to the data sparsity issue, in linguistics, words or their segmented-out lexemes are usually considered as basic units of meaning, which makes words to be more suitable when solving natural language processing tasks.

There are however two pressing issues here. The first issue is the absence of a perfect segmentation algorithm for any single language. A perfect segmentation algorithm should be able to segment given unsegmented sentence into a sequence of lexemes and morphemes. The other issue, which is specific to neural network approaches, is that neural machine translation systems suffers from increased complexity due to the large vocabulary size (Jean et al., 2015; Luong et al., 2015), which does not happen with character-level modelling.

Most issues of word-level modelling can be addressed to certain extent by switching into finer tokens, e.g., characters. In fact, to neural networks, each and every token in the vocabulary is treated as an independent entity, and the semantics of tokens are simply learned to maximize the objective function (Chung et al., 2016). This property allows a lot of freedom to the neural machine translation system in the choice of tokens.

The NYU-MILA neural machine translation system is built on the idea of directly generating characters, instead of words, that can possibly unlink a machine translation system from the need of explicit segmentation as a preprocessing step, which is often suboptimal in solving translation tasks. We focus on representing the target sentence as a sequence of characters, and the source sentence as a sequence of subwords (Sennrich et al., 2015).

2 System Description

In this section, we describe the details of the NYU-MILA neural machine translation system. In our system, we closely follow the neural machine translation model proposed by Bahdanau et al. (2015). A neural machine translation model (Forcada and Ñeco, 1997; Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014) 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

This system description paper summarizes and details the experimental procedure described in Chung et al. (2016)

Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers, pages 268–271, Berlin, Germany, August 11-12, 2016. ©2016 Association for Computational Linguistics

x_t and $y_{t'}$ are respectively source and target tokens. The neural network is constructed as a composite of an encoder network and a decoder network.

The encoder maps the input sentence X into its continuous representation. A bidirectional recurrent neural network, which consists of two recurrent neural networks (RNNs), is used to give more representational power to the encoder. 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 $\mathbf{z}_t = [\overrightarrow{\mathbf{z}}_t; \overleftarrow{\mathbf{z}}_t]$.

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: $\log p(Y \mid X) = \sum_{t'=1}^{T_y} \log p(y_{t'} \mid y_{<t'}, X)$. For each conditional term in the summation, the decoder RNN updates its hidden state by

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

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

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

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

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

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. 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:

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

The whole network, 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. In our system, the source sentence X is a sequence of subword tokens extracted by byte-pair-encoding (BPE) (Sennrich et al., 2015), and the target sentence Y is represented as a sequence of characters.

3 Experimental Settings

In this section, we describe the details of the experimental settings for our system.

Corpora and Preprocessing We use all available training parallel corpora for four language pairs from WMT'16: En-Cs, En-De, En-Ru and En-Fi. They consist of 63.5M, 4.5M, 2.3M and 2M sentence pairs, respectively. We do not use any monolingual corpus. We only use the sentence pairs, when the source side is up to 50 subword symbols long and the target side is up to 500 characters. For all the pairs other than En-Fi, we use newstest-2013 as a development set, and for En-Fi, we use newsdev-2015 as a development set.

All of the source corpora were preprocessed using BPE (Sennrich et al., 2015), and for the target corpora, no additional preprocessing step is required. For the target vocabulary, we use 300 characters and two additional tokens reserved for $\langle$EOS$\rangle$ and $\langle$UNK$\rangle$. For the source vocabulary we constrain the size of BPE symbols up to $30,000$.

Models and Training We use gated recurrent units (Cho et al., 2014) (GRUs) for the recurrent neural networks. The encoder has 512 hidden units for each direction (forward and reverse), and the decoder has two hidden layer with 1024 units each. The embedding layers of both source and target sides have dimensionality of 512 without any non-linearity. Both f_{out} and f_{score} are feedforward neural networks with an intermediate hidden layer with 512 $\tanh$ units.

We train the model using stochastic gradient descent with Adam (Kingma and Ba, 2014) using the default parameters introduced in the paper. Each update is computed using a minibatch of 128 sentence pairs. The norm of the gradient is rescaled with a threshold set to 1 (Pascanu et al., 2013). We set the initial learning rate of 0.0001.

Language Pair	BLEU-c	TER	Ranking		
			BLEU-c cons.	BLEU-c uncons.	Human
En-Cs	23.6	0.639	2/12	2/12	2/8
En-De	30.8	0.583	3/12	3/12	4/11
En-Fi	15.1	0.771	3/12	4/13	4/11
En-Ru	23.1	0.677	6/7	6/8	4/8

Table 1: Empirical results of the NYU-MILA systems on WMT'16 test sets. All of our submitted systems are constrained. For ranking by BLEU-c scores, when there are multiple submissions from a single system, we count it as one system. Some of the systems that showed in BLEU-c case do not show in the human evaluation, hence the total number of systems does not match. We present the ranking in both constrained setting (cons.) and unconstrained setting (uncons.) on the table.

Decoding and Evaluation We use beamsearch to approximately find the most likely translation given a source sentence. We use a beam width of 15 to find the model with best translation quality. The translation quality is evaluated by using BLEU.[1] For the WMT'16 test sets, we use the same beam width.

Ensembles We build an ensemble model using eight independent neural machine translation models initialized with different parameters. We decode from an ensemble by taking the average of the output probabilities at each step.

Decoding Speed of the Character-Level Decoder We evaluate the decoding speed of the character-level decoder and compare with a subword-level decoder on newstest-2013 corpus (En-De) with a single Titan X GPU. The subword-level decoder generates 31.9 words per second, and the character-level decoder generates 27.5 words per second. Note that this is evaluated in an online setting, where only one sentence is translated at a time, and translating in a batch setting could differ from these results.

4 Experimental Results

The results of the NYU-MILA system is presented in Table 1. The character-level decoding works well on most of the languages that are tested, achieving comparable BLEU-c scores to other approaches using words or subwords (BPE) as tokens. Note that our system does not incorporate extra monolingual training corpus, and does not include any kind of postprocessing e.g., reranking.

[1]We used the multi-bleu.perl script from Moses during training and internal evaluation.

5 Conclusion

We present the NYU-MILA neural machine translation system for WMT'16, which has a character-level decoder on the target side. Our results show that a character-level decoder can perform comparable to state-of-the-art systems. The NYU-MILA neural machine translation system achieved second rank in En-Cs and En-Fi (constrained only) and third rank in En-De. To the best of our knowledge the NYU-MILA system may be the only submitted system that directly generates characters instead of words or subwords. The biggest advantage of the character-level decoding approach is that the machine translation system no longer requires any preprocessing step, such as segmentation.

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)*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger

Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*.

Mikel L Forcada and Ramón P Ñeco. 1997. Recursive hetero-associative memories for translation. In *International Work-Conference on Artificial Neural Networks*, pages 453–462. Springer.

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*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*, volume 3, page 413.

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

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

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

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

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*.