ICLR 2018 REPRODUCIBILITY CHALLENGE: UNSUPERVISED NEURAL MACHINE TRANSLATION

Anthony Chen, Dheeru Dua & Robert Logan
Department of Computer Science
University of California, Irvine
Irivne, CA 92612, USA
{anthony.chen, ddua, rlogan}@uci.edu

ABSTRACT

We investigate the reproducibility of 'Unsupervised Neural Machine Translation' (Artetxe et al., 2017a). This paper introduces a novel architecture for neural machine translation which does not require parallel training corpora. In this report, we will provide commentary on how well this paper describes the experiments performed so that they can be replicated by other researchers, as well as provide an implementation of the model architecture introduced in the paper (available at https://github.com/rloganiv/monolingual-nmt). We give our implementation's results on a subset of the experiments conducted by the original authors. Overall, we find that we were not able to produce the same results as the original paper even after considerable effort.

1 INTRODUCTION

Machine translation is the task of designing a system capable of translating text from one language to another. With the recent advances in deep learning a significant progress has been made in neural machine translation (NMT). The introduction of sequence-to-sequence model and attention based decoding schemes (Bahdanau et al., 2014; Gehring et al., 2017; Vaswani et al., 2017; Wu et al., 2016) have led to significant improvement in the field of machine translation. These can be trained end-to-end unlike statistical machine translation (SMT) systems which require a careful deliberation in tuning the system.

However, as most neural systems, they need a lot of data and computation to achieve better results. Unfortunately, good quality parallel corpora can be extremely challenging and costly to build, especially in low-resource domains. In order to address this issue, Artetxe et al. (2017a) propose a neural architecture which can be trained in an entirely unsupervised manner using monolingual corpora by leveraging the linguist structural similarities in monolingual vector space embeddings. A significant amount of work has been done in learning word translations from one language to another in low-resource fields by exploiting bilingual word mapping dictionaries (Mikolov et al. (2013a); Smith et al. (2017); Artetxe et al. (2017b)). This work tries to take that to next level to perform unsupervised translation of sentences by leveraging the structure learned in word vector spaces.

In this report, we will evaluate how well this paper lends itself to being reproduced, particularly with respect to following criteria: the clarity of the model description, availability of datasets used in the paper, availability of code, availability of hyperparameters, computational requirements and the ease of re-implementing the model and reproducing experiment results. In addition we provide an implementation of the model, and study whether our implementation is able to reproduce a subset of experiments conducted in the paper.

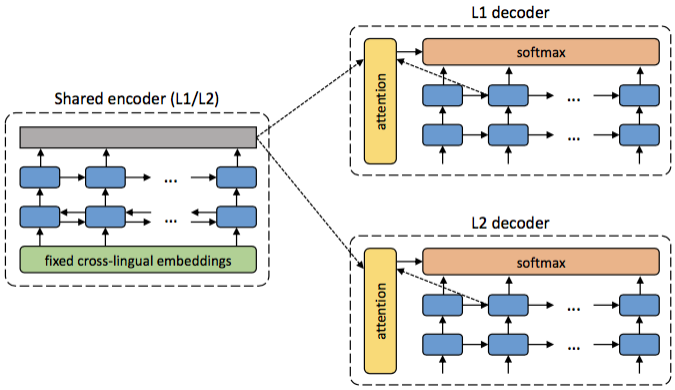

Figure 1: Model Architecture. Image taken from original paper. Each sentence is used in two ways. The first is *denoising*, which encodes a noisy version of the sentence and tries to optimize the reconstruction probability of the original sentence. The second is *backtranslation*, which first greedily translates a sentence from one language to the other. The translated sentence is then fed back through the encoder and tries to optimize the reconstruction probability of the original sentence.

2 Reproducibility Evaluation

2.1 Clarity of Model Description

Artetxe et al. (2017a) dedicate a total of four paragraphs as well as the figure presented in Figure 2.1 to describing their model architecture. The proposed model is described as consisting of two components: an encoder module which is shared by both source languages, and two language specific decoder modules. The authors note that they use a two-layer bidirectional GRU for the encoder, and a two-layer GRU for the decoder. In addition, the author's highlight the following aspects of their architecture:

1. The encoder is shared by both languages, which is made possible by using cross-lingual word embeddings. These cross-lingual embeddings are obtained by projecting monolingual embeddings from source and target language onto a common vector space. These embeddings are kept fixed during back-propagation to maintain stability. The code and hyper-parameters needed to create these word embeddings is described in detail in section 3.2. of the paper.
2. The decoders use the global attention mechanism with the general alignment function described by Luong et al. (2015).

We found this information to be sufficiently detailed and did not encounter any significant obstacles implementing the model architecture.

In addition, the authors dedicate almost an entire page to describing the training process, which considerably differs from the standard NMT training paradigm due to the lack of parallel data. This training process consists of two main phases: denoising and backtranslation. The denoising phase involves adding noise to the input sentence before encoding it, and then training the architecture to reconstruct the undistorted sentence. The authors add noise by swapping half of the words in a sentence. In the backtranslation phase, an input sentence is translated using greedy decoding into the other language. The system is then trained to translate this back to the original input sentence. The authors then proceed to detail how these training objectives are alternated between during training.

While the authors are very clear about the model architecture, some aspects are ambiguous. Firstly, while authors mention that dropout is applied during training, they do not specify where it is applied (e.g between hidden layers, before the decoder output, etc.). Secondly, the authors also did not specify whether the fixed cross-lingual embeddings were also used in the decoder layers. Although

Table 1: Time spent implementing each component.

Component	Implementation Time (in days)
Data Processing	2
Cross-lingual embeddings	0
Model architecture	10
Denoising	1
Backtranslation	4
Beam search	3
Total time	**20**

these are only minor oversights, we feel that in order for the work to be properly replicated such details should be included.

2.2 Availability of Datasets

The News Crawl corpus with articles between 2007 and 2013 is used as a source of unsupervised training data. Benchmarking is performed on the French-English and German-English datasets from the WMT 2014 shared task, which is standard for NMT tasks. Both of these datasets are publicly available and a link was provided in the paper, however the datasets require some preprocessing in order to be used (see Section 2.4).

2.3 Availability of Hyperparameters

The authors spend a considerable amount of space detailing the various hyperparameters used in modeling, training, and testing. We reiterate the hyperaparameters here.

Word embeddings for each language are obtained using a skip-gram model with ten negative samples, 300 dimensions, a context size of ten words, a sub-sampling rate of 10^{-5}, and trained for ten iterations (Mikolov et al., 2013b). These embeddings are then used to create the cross-lingual word embeddings with embedding size 300 using the hyperparameters in Artetxe et al. (2017a). The authors also mention that they test applying byte-pair encoding (Sennrich et al., 2016).

The encoder and both decoders consist of two layer GRUs with 600 hidden units (Cho et al., 2014). The attention used is global attention with an alignment function (Luong et al., 2015).

Training is done using the Adam optimizer with a learning rate of $\alpha = 0.0002$ and a batch size of 50 (Kingma & Ba, 2014). Across models are trained through 300,000 iterations. During training, dropout is applied with a probability of 0.3 (Srivastava et al., 2014). During backtranslation, decoding is done greedily, meaning a beam search of width 1. During test time, translations are produced using beam search with a beam-width of 12.

We find that the hyperparameters described are exhaustive. No additional parameters needed specification in order to conduct the experiments.

2.4 Availability of Code

Data preprocessing requires tokenizing and truecasing the input data. The authors note that this can be done using the Moses toolkit. Furthermore, cross-lingual word embeddings are needed for both languages. These are produced by first training skip-gram embeddings for each language, then use the public implementation of the method proposed by Artetxe et al. (2017b) to map these embeddings into a shared space. All of the necessary links needed to carry out these preprocessing steps were provided in the paper. However, the model code is not made publicly available.

2.5 Computational Requirements

In the paper, it is directly stated that all models were trained for 300,000 iterations. The authors note this takes between 4-5 days to complete on a Titan X GPU for the full unsupervised model. Thus

Table 2: Quantitative Results Comparison. Each entry has two values. The first value represents our final results, the second is the value in the original paper.

		FR-EN	EN-FR	DE-EN	EN-DE
Unsupervised	1. Baseline (nearest neighbor)	- / 9.98	- / 6.25	- / 7.07	- / 4.39
	2. Proposed (denoising)	**5.68** / 7.28	**2.49** / 5.33	- / 3.64	- / 2.40
	3. Proposed (+ backtranslation)	**7.82**/15.56	**2.62**/15.13	- / 10.21	- / 6.55
	4. Proposed (+ BPE)	- / 15.56	- / 14.36	- / 10.16	- / 6.89
Semi-supervised	5. Proposed (full) + 100k parallel	- / 21.81	- / 21.74	- / 15.24	- / 10.95
Supervised	6. Comparable NMT	- / 20.48	- / 19.89	- / 15.04	- / 11.05
	7. GNMT	- / -	- / 38.95	- / -	- / 24.61

the authors are fully transparent about the amount of computation needed to train the model both in terms of hardware (e.g. it can be done on a single high-performance GPU) as well as how much computing time is required. Although we are able to access similarly powerful hardware, due to the long training times for these models, we were only able to conduct a fraction of the experiments conducted in the original paper. Further details are provided in Section 3.

2.6 Ease of Re-Implementation

Our implementation of the model uses PyTorch[1], which is the same library used by the authors. To further facilitate the task, we use the PyTorch based OpenNMT[2] library which provides encoder and decoder classes which can be adapted to fit the model specifications with minimal effort, as well as a suite of useful functions for applying attention and performing beam search (Klein et al., 2017).

While hyper-parameters and model specifications were clearly stated, we still ran into several difficulties during re-implementation process. In the early stages of development we attempted to build the model without depending on OpenNMT and expended a lot of time on developing the model architecture. The basic architecture was not too difficult to surmise, but required quite a bit of modules, which became difficult to work with when we needed to send information from the encoder to the decoder.

While implementing the model architecture, we encountered a few issues. Firstly, the paper does not provide information on how the initialization of the two respective layers in the decoders. Based on our assumptions, we initialize the first layer of decoder with the encoded representations and the second layer with random values.

Secondly, The references on backtranslations[Sennrich et al. (2015)] do not update the encoder and attention weights, however, from our understanding of the paper the attention and encoder weights are updated during backtranslation.

We provide quantitative measurements of the time taken to implement each component in Table 1.

3 Reproducing Experiments

As mentioned in Section 2.5, although our hardware was sufficiently powerful to train the model, we lacked the time to be able to carry out the entire suite of experiments conducted in the paper. Accordingly, we focused on what we found to be the key contribution: unsupervised training with denoising and backtranslation. Specifically, we attempted to replicate the FR-EN and EN-FR results in lines 2 and 3 of Table 2.

Following the paper, we train models for 300,000 iterations. For the denoising model, a single training iteration consists of one batch of denoising English sentences and one batch of denoising French sentences. For backtranslation, a single training iteration consists of performing the previously mentioned denoising updates, as well as backtranslation updates for each language. Concerning the ambiguities raised in Section 2.1 we chose to use the fixed embeddings in the decoder, and to

[1] `http://www.pytorch.org`

[2] `https://github.com/OpenNMT/OpenNMT-py`

Training Iteration	**FR-EN**	**EN-FR**
150k	1.62	10.42
200k	9.99	12.88
250k	8.01	2.83
300k	7.82	2.62

Table 3: Backtranslation BLEU scores at different points during training

Source Sentence	**Gold Translation**	**Model Output**
A black box in your car?	Une boîte noire dans votre voiture?	Un passage noir dans votre voiture?
They can choose a device with or without GPS.	Ils peuvent choisir un appareil avec ou sans GPS.	Ils peuvent choisir un appareil avec ou sans GPS.
La taxe fédérale elle-même, qui est de 18,4 cents par gallon, n'a pas augmenté depuis 20 ans.	The federal tax itself, 18.4 cents per gallon, hasn't gone up in 20 years.	The Federal tax itself , which is about $ 6.9 million by gallon , has not increased over 20 years.
L'annonce de la parution d'apos; un nouvel album de David Bowie avait laissé tout le monde pantois.	The announcement that David Bowie was releasing a new album had stunned the world.	Last week, the publication of a new album of David Bowie had left it, but it was the world..

Table 4: Example Model Outputs

use the standard dropout implementation in OpenNMT. Our full implementation can be found at: `https://github.com/rloganiv/monolingual-nmt`.

3.1 Results

Our results are displayed alongside the original paper results in Table 2. As is readily observed, we were unable to reproduce the results for the both the denoising model and the backtranslation model. In fact, the our final BLEU scores are well below the baseline nearest neighbor method. The disparity between our results is difficult to explain, especially given how well the model and its parameters were documented. One potential factor may be our decision to use fixed embeddings in the decoder layer. Another potential explanation is that the model may have difficulties converging during training, and is sensitive to initialization. To support this, we observe that BLEU scores varied drastically over the course of training, as is shown in Table 3.1. We obtained much closer scores to the one's provided in the paper around the 200,000th training iteration. We would be interested to know if the main authors saw similar instability during their model training.

It is also worth noting that despite our apparently terrible BLEU scores, a qualitative analysis of the model outputs shows that the model didn't totally fail to learn how to translate. Some example outputs of our model are provided in Table 3.1. Observe

4 Conclusions

We experimented with few different variations of the architecture, however were unable to replicate the BLEU scores reported by the authors. In the initial iterations the BLEU scores improve drastically and reach close to the results in the paper, however, on further training the results start to regress considerably. We observe that the training is unstable as more data is seen by the model. The semi-supervised setting would perhaps give better results but we did not have enough resource to execute those experiments.

Despite our failure to reproduce the experimental results, we would like to recognize all of the effort that the authors put in to making their work reproducible: they used widely available datasets for training and evaluation, they described how data preprocessing should be performed and gave links to all of the tools that they used, and meticulously detailed all of the hyper-parameters that they used.

ACKNOWLEDGMENTS

We would like to thank the authors for quickly responding to our questions as well as providing us with pre-trained cross-lingual word embeddings.

REFERENCES

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Learning bilingual word embeddings with (almost) no bilingual data. pp. 451–462, 2017a.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 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)*, pp. 451–462, 2017b.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *Proceedings of the 2014 International Conference on Learning Representations*, 2014.

Kyunghyun Cho, Bart van Merrienboer, and Çaglar Gülçehre and Dzmitry Bahdanau and Fethi Bougares and Holger Schwenk and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. 2014.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. Convolutional sequence to sequence learning. 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations*, 2014.

Guillaume Klein, Yoon Kyung Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation. 2017.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. 2015.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. 2013b.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *Proceedings of the 54th Annual Meeting of the Association for Compu- tational Linguistics*, 2016.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*, 2017.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint*, arXiv:1609.08144, 2016.