Round Trip Loss for Machine Translation

Anja Adamov[a]*, Lauro Böni[b]†, Simon A. Broda[cd]‡, Urs Vögeli[b]§

[a] *IBM Switzerland Ltd., Zurich, Switzerland*

[b] *ETH Zurich*

[c] *Department of Banking and Finance, University of Zurich*

[d] *Quantitative Economics Section, University of Amsterdam*

January 17, 2020

Abstract

We augment a machine translation system with a round-trip loss to encourage the system to generate translations that, when translated back into the source language, retain much of the original structure. We surmised that in doing so, the model would learn internal representations with improved semantic meaning. Alas, this conjecture was not borne out by the results; our experiments show that incorporating the proposed loss has little to no effect on the obtained translations in terms of their BLEU scores.

Key Words: Cycle consistency loss; deep learning; natural language understanding; machine translation; round-trip loss.

* *E-mail address:* adamova@student.ethz.ch
† *E-mail address:* laboeni@gmail.com
‡ *E-mail address:* simon.broda@uzh.ch
§ *E-mail address:* voegeli.urs@gmail.com

1 Introduction

Machine translation is an important task within the field of natural language understanding. Consequently, a variety of models have been proposed for solving it. One of the first truly successful models was the sequence-to-sequence (seq2seq) model of Sutskever et al. (2014), while the current state of the art builds upon the *Transformer* architecture introduced by Vaswani et al. (2017). At a high level, both the seq2seq and the Transformer architectures are comprised of an encoder and decoder; the encoder learns an internal representation of the source sentence, and the decoder decodes it into the target language. For these models to work, the internal representation must capture the semantic meaning of the source sentence.

Irrespective of the architecture, these models are typically trained with the cross-entropy loss between the ground truth and predicted sentences, usually with teacher forcing. Here, we propose to add a round-trip penalty to the loss function of the model. The idea is that instead of training a single model to translate from a source language $\mathcal{S}$ to a target language $\mathcal{T}$, one trains two models (of identical structure), one mapping $\mathcal{S} \mapsto \mathcal{T}$ and one mapping $\mathcal{T} \mapsto \mathcal{S}$. The two models are, at first, trained independently and in parallel, by feeding in the same batches of sentence pairs (in this paper, we rely on the Transformer architecture of Vaswani et al. (2017), but the idea applies to any encoder-decoder architecture). After τ epochs, the models are then trained jointly, using as loss a sum of four cross-entropy terms, viz.,

$$\mathcal{L} = CE(s, \hat{s}) + CE(t, \hat{t}) + \lambda \left(CE(s, \tilde{s}) + CE(t, \tilde{t}) \right). \tag{1}$$

Here, $s \in \mathcal{S}$ is the ground truth sentence in the first language, $t \in \mathcal{T}$ is the corresponding ground truth in the second language, $\hat{s} \in \mathcal{S}$ and $\hat{t} \in \mathcal{T}$ are the respective translations of t and s, $\tilde{s}$ is obtained by translating $\hat{t}$ back to $\mathcal{S}$, and $\tilde{t}$ is obtained by translating $\hat{s}$ back to $\mathcal{T}$. τ and λ are hyperparameters.

The round-trip loss is inspired by the cycle consistency loss pioneered by Zhu et al. (2017) in the context of GANs. The only application of this concept to the field of machine translation of which we are aware is Su et al. (2018), who adapt it for unsupervised multi-modal machine translation. Here, we propose to apply the idea to supervised machine translation. We conjecture that encouraging the model to generate round-trippable translations will help it learn a semantically meaningful representation. A related idea is that of back-translation, pioneered by Sennrich et al. (2016). Sennrich et al. propose to augment the parallel training corpus used to train a machine translation system with synthetic data, obtained by translating a monolingual corpus in the target language to the source language using an independently trained system. They argue that this is beneficial in particular when parallel training data is scarce. The difference with our approach is two-fold: i) in our approach, the two networks are trained *jointly*, each with their own round-trip loss term, whereas in back-translation, the models are trained separately, with only one of them benefiting from back-translation; ii) we do not assume the existence of a separate monolingual corpus in the target language, but work strictly with a parallel corpus. Having ground truth available, we are thus able to use teacher forcing in constructing the round-trip loss contribution, greatly speeding up training.

The remainder of this manuscript is organized as follows. Section 2 describes the model architecture and our implementation of it. Section 3 details the results of our experiments with the proposed round-trip loss. Section 4 provides a discussion and concludes.

2 Models and Methods

2.1 Model Architecture

This section is, unless explicitly stated otherwise, based on Vaswani et al. (2017). Useful background information on Recurrent and Convolutional Neural Networks is based on Goodfellow et al. (2016).

The core idea behind the classical Transformer model is the so-called self-attention — the model's ability to attend to different positions of the input sequence to compute a representation of that sequence - without imposing any recurrence nor convolution to the model.

For deep neural networks aiming to perform sequence translation, some general sort of memory is required as, in practice, a lot of sequences depend on words from previous sequences. As an example for the reader, try to complete the phrases ”My father is a winegrower. He likes to drink **. . .** ”. In this case, you might guess the missing word ”wine” correctly - with the help of attention. And this is exactly, what modern natural language models try to capture.

Recurrent Neural Networks (short: RNN) tackle this problem by a specific architecture that focuses on time-dependencies. However, they are hardly or even not parallelizable and hence computationally suboptimal. On the other hand, Convolution Neural Networks (short: CNN) are easily parallelizable and can exploit local dependencies. However, these models do not explicitly rely on the attention.

The Transformer model solely relies on attention - more precisely on self-attention - to capture global dependencies between input and output. In a nutshell, the model consists of a set of encoders (EC) and set of decoders (DC). The original model in (Vaswani et al. (2017)) uses $N = 6$, i.e. 6-fold stacking of the respective ECs and DCs. Each EC maps input sequences $(x_1, ..., x_n)$ to its continuous representation, say, $(z_1, ..., z_n)$ which in turn are used by DCs to generate the output respective the translated prediction $(y_1, ..., y_m)$.

More precisely, each EC consists of two components: self-attention and feed-forward neural networks (FNN). The self-attention mechanism takes in a set of input encodings of length $d_{\text{model}} = 512$ from the previous EC, splits it up to a key, value and query vector (each of length $d = 64$) and then tries to learn their relevance to each other to generate a set of output encodings, again of length d_{model}. The FNN then further processes each output encoding individually (the dimensionality of the input and output is therefore d_{model}, whereas the size of the inner-layer

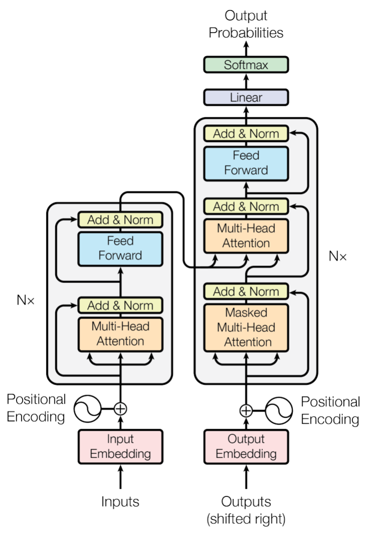

Figure 1: Architecture of the Transformer model from Vaswani et al. (2017).

is $d_{\text{ff}} = 2048$). The output encodings are then finally passed to the next EC as its input, as well as to the DC.

On the other hand, each DC consists of three components: self-attention, an attention mechanism over the encodings and a classical FNN. The first EC takes positional information and embeddings of the input sequence as its input, rather than encodings. The positional information is necessary for the Transformer to make use of the order of the sequence, as it does not rely on recurrence nor convolution. Each DC functions in a similar fashion to the ECs, but an additional (masked) attention mechanism is inserted which draws relevant information from the encodings generated by the EC. Finally, the last DC is then followed by a linear transformation and a softmax layer to produce the desired output probabilities.

This somehow abstract description is visualized in Figure 1. Moreover, Figure 2 shows a visualization of the self-attention mechanism in the ECs where the relevance of each embedding (i.e. word) are compared.

2.2 Implementation and Training Details

We use the Transformer implementation in TensorFlow 2.0 available from the tensorflow website and implement a custom training loop, which instantiates the $\mathcal{S} \mapsto \mathcal{T}$ and $\mathcal{T} \mapsto \mathcal{S}$ models and trains them jointly with the loss described in (1). We train our models on the WMT'14 English-German data set available from https://nlp.stanford.edu/projects/nmt/. The training set consists of 4,508,785 sentence pairs, of which we retain those where source and target sequence both have a length of 40 or fewer tokens to keep the computational demands manageable (the original Transformer model in Vaswani et al. (2017) uses a maximal sequence length of 65). We employ a subword tokenizer, and experiment with relatively

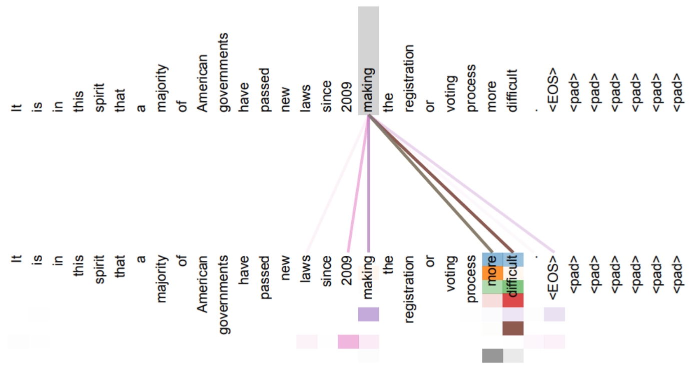

Figure 2: Visualization of the self-attention mechanism from Vaswani et al. (2017).

small target vocabulary sizes of both 2^{13} and 2^{14}, again to keep computational demands at bay (a production system would likely use a vocabulary at least twice as large). Compared to the base transformer, we also reduce the number of layers from from 6 to 4, d_{model} from 512 to 128, and d_{ff} from 2048 to 512. The number of attention heads is kept at 8. With these choices, training the model (which consists of two transformers) for a single epoch on the entire dataset takes about 72 minutes on a single NVIDIA GTX1080 Ti. We use the learning rate schedule that was used for the original Transformer model in Vaswani et al. (2017).

At test time, we rely on the beam search decoder implementation from tensor2tensor. We use a beam width of 10 and a length penalty of $\alpha = 0.65$, per the recommendations of Wu et al. (2016). We assess the quality of the translations of the 3,003 sentences in the test set using the BLEU score (Papineni et al., 2002), computed using the nltk package.

3 Results

3.1 Experiments

In order to evaluate the effect of our proposed round-trip loss, we conduct a number of experiments. First, we train and evaluate the model from scratch for different values of λ. This corresponds to setting $\tau = 0$ in (1). The results after training each model for 15 epochs are shown in Table 1.

λ	0.0	0.02	0.05	0.1	0.2	0.3	0.4	0.6
$\nu = 2^{13}$	11.69	11.00	10.34	**11.89**	10.65	9.75	11.29	11.19
$\nu = 2^{14}$	**12.38**	11.81	12.04	12.11	12.23	12.15	11.80	11.68

Table 1: BLEU scores for models trained from scratch, for different target vocabulary sizes ν.

The results are, at best, inconclusive. For vocabulary size $\nu = 2^{13}$, while the best BLEU score is obtained with $\lambda = .1$, there is very little variation in the resulting BLEU scores within the range of λ values tested. With vocabulary size $\nu = 2^{14}$, the picture is similar, although the best score is now obtained with $\lambda = 0$ (i.e., no round trip loss). While we have not formally tested this, the variation in scores appears small enough to be entirely attributable to random fluctuations.

In order to verify that the round-trip loss has any effect at all (and to rule out an error in our implementation), we trained a model with the unreasonably large value of $\lambda = 100$. This resulted in a BLEU score of less than 0.1, confirming that the round-trip loss does eventually have an effect on the results, albeit negative in this case.

In our next set of experiments, we pre-train a model without round-trip loss ($\lambda = 0$), and then fine-tune it for another 15 epochs using the same values for λ as before. This corresponds to a non-zero value of τ in (1). As before, we use both 2^{13} and 2^{14} as our vocabulary size, and set $\tau = 12$ for the former and $\tau = 15$ for the latter. For comparison, we also trained a model with $\lambda = 0$ for $\tau + 15$ epochs. The results are shown in Table 2.

λ	0.0	0.02	0.05	0.1	0.2	0.3	0.4	0.6
$\nu = 2^{13}$	**12.24**	12.05	11.72	11.83	11.95	12.07	11.94	11.67
$\nu = 2^{14}$	**13.27**	11.52	12.62	12.86	12.71	12.59	12.60	12.44

Table 2: BLEU scores for pre-trained models after fine tuning, for different target vocabulary sizes ν.

The results are qualitatively similar to those without pre-training, except that the best score is now obtained with $\lambda = 0$ for both vocabulary sizes. As before, what little difference exist between models appears attributable to random variation.

Finally, we conducted a last set of experiments in which we aimed to compare our approach to the back-translation approach of Sennrich et al. (2016). Note that even when trained without round-trip loss ($\lambda = 0$), our implementation always trains two models jointly, one for translating $\mathcal{S} \mapsto \mathcal{T}$ and one for $\mathcal{T} \mapsto \mathcal{S}$. Thus, after training on a subset of the data for 15 epochs, we used the model to translate part of the remaining training data from $\mathcal{T} \mapsto \mathcal{S}$, and then augmented the training data in the source language with these synthetic data. We then trained a model from scratch on this augmented data set and evaluated it as usual. Unfortunately, these experiments were heavily constrained by computational time, mostly because translation is very costly: our implementation will translate roughly 3000 sentences per hour. As such, we decided to back-translate only around 0.5% (21,600 sentences) of the training data. To allow this small augmentation to make a difference, we restricted the ground truth part of the data to 10% of its original size. We also restrained ourselves to a tokenizer with a target vocabulary size of 2^{13} to further reduce computational cost.[1] With these restrictions, all models (with and without

[1]We experimented both with using the original tokenizer trained on the entire dataset, and a

round trip loss) resulted in BLEU scores below 1. These do not allow for a proper comparison and are thus not reported here.

3.2 Example Translations

The following are examples for $\lambda = 0.1$ with a target vocabulary size of 2^{13} and $\tau = 12$.
Input: ”Er lief auf Sand die gesamte Länge der Arena und obwohl er so winzig aussah, hat er phantastische Arbeit geleistet.”
Predicted translation: ”He loved the entire length of the Arena and even though he was so wing, he has done fantastic work.”
Input: ”Kritiker wie die demokratische Senatorin Claire McCaskill aus Missouri vertreten die Ansicht, es gebe keine gültigen Sicherheitsgründe für das Verbot.”
Predicted translation: ”Critics such as the democratic Senator Claire McCaskill for Missouri represent the view that there are no valid security reasons for the ban.”

4 Discussion and Conclusion

In this work, we proposed to augment a neural machine translation system with a round-trip loss, inspired by cycle consistency loss of Zhu et al. (2017). In our experiments, we were unable to demonstrate any effect of the additional loss term, positive or negative, for a range of weights. We did, however, find that the additional loss term does have an effect when given extremely large weight. As such, it is conceivable that further experimentation with moderately large weights might find more encouraging results.

These results notwithstanding, we maintain that some form of regularization towards round-trippable translations should be able to improve translation quality, both because of the strong intuitive appeal of the idea, and because of its relation to the rather successful concept of backtranslation. An alternative to the approach pursued herein would be to add a regularization term that penalizes the squared norm of the difference between the outputs of the encoders of the two models $\mathcal{S} \mapsto \mathcal{T}$ and $\mathcal{T} \mapsto \mathcal{S}$. We leave these experiments to subsequent work.

References

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`. 2

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL ’02. 4

tokenizer trained on only the subset used for training.

Sennrich, R., Haddow, B., and Birch, A. (2016). Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics. 1, 5

Su, Y., Fan, K., Bach, N., Kuo, C. C. J., and Huang, F. (2018). Unsupervised multimodal neural machine translation. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. 1

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14. 1

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17. 1, 2, 3, 4

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Łukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. 4

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*. 1, 6