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 5, 2020

Abstract

We propose to augment a machine translation system by adding a round-trip loss which encourages the system to generate translations that when translated back into the source language, retain much of the original structure. We show that in doing so, the model will learn internal representations with improved semantic meaning.

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 par-

allel 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 Section 5 concludes.

2 Models and Methods

2.1 Model Architecture

TODO: explain what d_{ff}, d_{model} are so we can refer to it later As described in previous sections, we have based our model extension on the transformer model implementation at `cite`. 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. (ref: https://www.tensorflow.org/tutorials/text/transformer)

For deep neural networks performing 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 focus on the attention part.

The transformer model is a combination of a CNN together with attention - more precisely with self-attention. In a nutshell, the transformer model consists of a set of encoders (EC) and set of decoders (DC). The ECs map input sequences $(x_1, ..., x_n)$ to its continuous representation, say, $(z_1, ..., z_n)$ which in turn is 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 from the previous EC and weighs their relevance to each other to generate a set of output encodings. The FNN then further processes each output encoding individually. These output encodings are finally passed to the next EC as its input, as well as to the DC. **cite wikipedia**

On the other hand, each DC consists of three components: self-attention, an attention mechanism over the encodings and a 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. Each DC functions in a similar fashion to the ECs, but

an additional attention mechanism is inserted which instead draws relevant information from the encodings generated by the EC. Like the first EC, the first DC takes positional information and embeddings of the output sequence as its input, rather than embeddings. The last DC is followed by a final linear transformation and softmax layer, to produce the desired output probabilities.

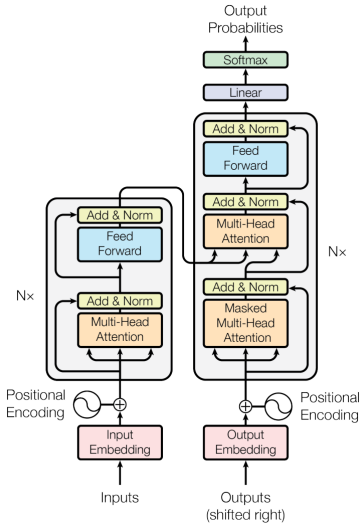

Figure 1: Architecture of the Transformer model

This somehow abstract description is visualized in Figure 1, where the original model in (**cite**) uses $N = 6$, i.e. 6-fold stacking of the respective ECs and DCs.

The self-attention part in both the EC as well as DC consists of a quite involved procedure that, in a nutshell, learns contextual dependencies. The authors of **cite** visualized the attention part by Figure 2:

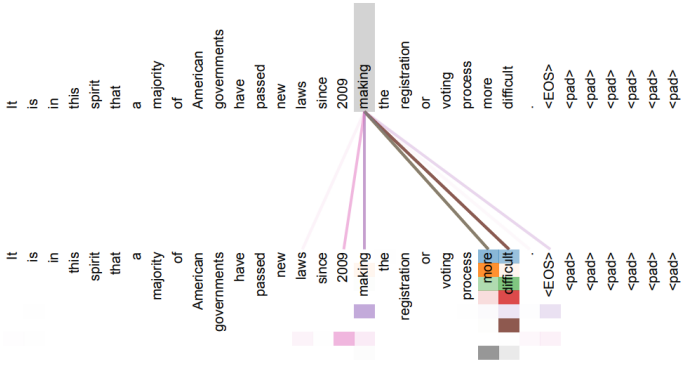

Figure 2: Visualization of the self-attention mechanism.

For a detailed description of the attention mechanism, we refer to **cite** (original

paper) or some interesting url.

to-be-continued cite:http://jalammar.github.io/illustrated-transformer/
Wikipedia
https://towardsdatascience.com/transformers-141e32e69591

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 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 ... on a single NVIDIA GTX1080 (?). We use a learning rate schedule that increases the learning rate from ... to ... over ... warm-up periods, and then decreases it again, down to a final value of ...

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). The BLEU score for the 3,003 sentences in the test set is computed using the nltk package.

TODO: implementation of backtranslation

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 ...

Next, we pre-train a model with $\lambda = 0$ for 15 epochs, and then fine-tune it for another 15 epochs using the same values for λ as before. The results are shown in Table ...

Finally, in order to compare our approach to the back-translation approach of Sennrich et al. (2016), ...

3.2 Example Translations

4 Discussion

Discuss the strengths and weaknesses of your approach, based on the results. Point out the implications of your novel idea on the application concerned.

5 Summary

Summarize your contributions in light of the new results.

References

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, 4

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, 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 networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*. 1