TECHNICAL UNIVERSITY OF DENMARK

02456

DEEP LEARNING

Neural Machine Translation

Authors:	*Study numbers:*
Christoffer ØHRSTRØM	s182827
Gabriel K. BROHOLM	s144454
Oldouz MAJIDI	s163502

January 3, 2019

1. ABSTRACT

In this work we seek to understand and build upon modern approaches to neural machine translation. In order to gain this understanding, we implement a model using local attention and compare it to a similar model without any attention mechanism. Training and evaluation are done on separate parts of the Multi30K dataset. We find that the model with local attention achieves a BLEU of 30.37 which greatly outperforms the model without any attention mechanism with a BLEU of 16.62. The BLEU of 30.37 is close to state of the art models but not higher.

2. INTRODUCTION

Neural Machine Translation (NMT) has in the span of the last decade become increasingly efficient in terms of time and accuracy. This development allows for greater communication across people, countries and cultures and is of both academic and practical interest. In this paper we seek to understand how modern NMT works and to build upon modern NMT models.

The accuracy of our model will be measured using BLEU [1] which is a metric to evaluate the quality of an translation and it also works independently of source and target language. The experiments on the model are carried out on the training and validation parts of the Multi30K dataset [2] and the BLEU score is tested on the test part of the same dataset.

This paper will start in section 3 by referencing the related works by giving a brief outline of the different NMT models. In section 4 our model for this project is described together with a simpler model, followed in section 5 by the experimental setup of the model. In section 6 we present the BLEU of our model as well as compare it to the simpler model and models by other authors using the Multi30K dataset.

3. RELATED WORKS

Traditionally machine translation systems rely on a component based design. This approach is often implemented in statistical machine translation (SMT) systems. NMT have both been used as a component in a SMT system [3] and as a standalone end-to-end system. In this work we implement NMT as a standalone end-to-end system. An example of doing so is the sequence to sequence model [4] by Sutskever et al. The authors proposed a network architecture where an encoder transforms a source sequence (sentence) into a vector which the decoder uses to make target sequences (translations). This approach is useful but the decoder is constrained to generate translations from the single fixed size vector generated by the encoder. Bahdanau et al. [5] argued that this creates a bottleneck and solved the issue by proposing an attention mechanism that enables the model to learn which part of the source sequence to pay attention to during decoding. This, in turn, enabled the model to more reliably translate long sentences. Our model is a sequence to sequence model using an attention mechanism but we are using a variant of attention that was proposed by Luong et al. [6]. This variation is called local attention and the main difference between local attention and the attention mechanism of Bahdanau et al. is that local attention constrains itself to only use a subset of the source sentence.

4. MODEL

Our model seeks to generate the most likely translation y given a source sentence x. In other words we want to maximize the probability $p(y|x)$. We do this by using a sequence-to-sequence model [4] with a local attention mechanism [6]. Let $x = (x_1, \ldots, x_{T_x})$ be a source sentence of length T_x and let $y = (y_1, \ldots, y_{T_y})$ be a translation of x of length T_y. The conditional probability our model seeks to maximize can then be expressed as

$$\begin{aligned} p(y|x) &= \prod_{j=1}^{T_y} p(y_j|y_{j-1}, \ldots, y_1, x) \\ &= \prod_{j=1}^{T_y} g(y_{j-1}, s_{j-1}, c_{j-1}) \end{aligned} \tag{1}$$

where g is some non-linear function that computes a probability distribution over the vocabulary of the target language, s_j is the hidden state of the decoder at time j and c_j is the context computed by the attention mechanism at time j. We note that we are making a Markov assumption, meaning that we assume that any subsequent word in a translation can be predicted from the information from the previous timestep.

Figure 1 shows the model architecture. Section 4.1 describes how the encoder and decoder works. It also describes how the encoder-decoder approach can be used to make predictions without an attention mechanism. This approach is compared in section 6 with the model displayed in figure 1. Section 4.2 goes in to detail on how the local attention mechanism works.

4.1. Encoder and Decoder

Both the encoder and decoder are recurrent neural networks (RNN). Specifically we use a LSTM [7] architecture to make the RNNs more resilient to long sequences. That is we use LSTM cells to make the encoder and decoder able to remember the sequence history. For the encoder that entails previous words in the input sentence and for the decoder it is the previously generated words in the translation. Models using this encoder-decoder architecture [3,4] generate a single fixed sized context vector, c, as the output of the encoder which is then given as input to the decoder. Let $e = (e_1, \ldots, e_{T_x})$ be the T_x

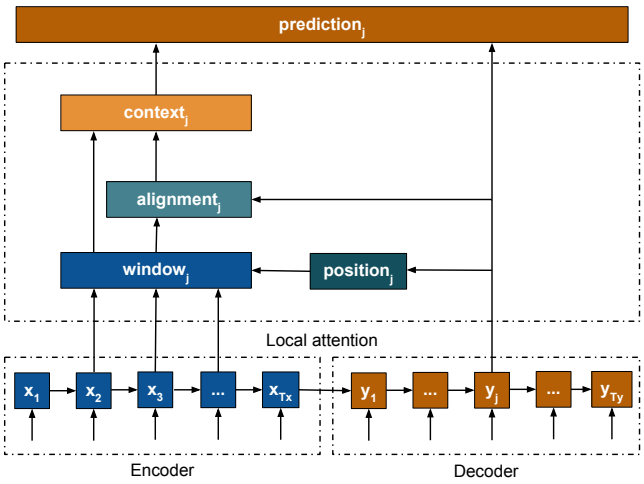


Fig. 1: Model architecture. Encoder and decoder uses LSTM to process source sentence and translation. Translated words are generated through a local attention mechanism that uses the encoder and decoder outputs. See section 4.1 and 4.2 for a detailed explanation of the architecture.

outputs of the encoder. We then compute the context c as

$$c = e_{T_x} \tag{2}$$

similarly to [4]. We note that there are several valid methods for how to compute c from e. The decoder will at this point use c (and the previously translated word and decoder hidden state) to translate one word at a time. This process continues until the decoder outputs the special token "EOS" (end of sequence).

4.2. Attention Mechanism

We will now describe how local attention works. The process can be seen in the center dashed box in figure 1 and the details will be explained in the following: Let $d = (d_1, \ldots, d_{T_y})$ be the T_y outputs of the decoder and consider the situation that we want to generate the jth translated word. The first thing local attention seeks to do, is to use a feed-forward network to find a position in the source sentence to place a window around. This is expressed in equation 3 where a_p is a feed forward network and σ is the sigmoid function.

$$\text{position}_j = p_j = T_x \cdot \sigma(a_p(d_j)) \tag{3}$$

Note especially that $p_j \in [0; T_x]$. Local attention then centers a window of size w around p_j such that the window contains the $2w + 1$ outputs of the encoder that lie within the window.

$$\text{window}_j = (e_{p_j - w}, \ldots, e_{p_j}, \ldots, e_{p_j + w}) \tag{4}$$

The parameter w is a hyperparameter and so it must be set empirically. The window can now be used to compute an alignment of the encoder outputs and d_j. The interpretation of this alignment is as a scoring of where to pay attention to in the source sentence where we restrict ourselves to the local area in the window. The alignment requires a scoring function to be computed. Luong et al. [6] generally find the dot product of d_j and window$_j$ to be optimal and therefore we also use this as a score function. The output of the score function is then transformed into a probability distribution by a softmax. Finally the probability distribution is multiplied by a gaussian centered around p_j with a standard deviation of $w/2$. This is done to enforce that words close to p_j are important to pay attention to. The alignment can be computed as in equation 5.

$$\text{alignment}_j = a_j = \text{softmax}(d_j^T \cdot e) \cdot \text{gaussian}(p_j, w/2) \tag{5}$$

The context vector, c_j, can now be computed as a weighted average of the window with weights given by a_j. This can be seen in equation 6.

$$\text{context}_j = c_j = \sum_{i=1}^{2w+1} a_{j_i} \cdot \text{window}_{j_i} \tag{6}$$

In order to predict the jth word in the translation, the model will concatenate d_j with c_j and pass the concatenated output through a feed forward network where the output is transformed into a probability distribution using a softmax.

This concludes the description of our model. To briefly summarize: We are using an encoder-decoder LSTM and a local attention mechanism.

5. EXPERIMENTAL SETUP

We train the network on the Multi30K dataset [2] and use a separate part of the dataset for evaluating the BLEU score. Multi30K contains around 30,000 pairs of translations from English to German of image captions from flickr.com. Furthermore, we train and evaluate two models: One model does not use any attention and is described in section 4.1. We call this model "S2S" for sequence-to-sequence. The other model uses local attention and therefore we will refer to this model as "LA". "LA" is described in section 4.2. We experiment with two models in order to see what effect local attention has on the quality of translations.

5.1. Training Details

The two models are both trained using the same training and validation set and they are trained using a Tesla K40c GPU. The implementation of the models is built using PyTorch [8] version 0.41 and the loss is computed using the cross entropy loss. The loss is minimized using the Adam [9] optimizer.

Both of the models depend on previous outputs when generating a word. Often, it can be beneficial for recurrent neural networks, with this kind of dependency, to apply a level of

teacher forcing [10]. We apply a variant of teacher forcing during training of both models. That is, with a probability p we feed the correct input to the decoder instead of the previous output from the decoder. Empirically we have found that $p = 0.75$ yields good results on the validation set. Likewise we have performed different experiments to optimize the hyperparameters of the models against the validation set. The result of those experiments can be seen in table 1 where we see the chosen hyperparameters of the models.

Hyperparameter	**Value**
Batch Size	128
Epochs	14
Learning Rate	0.003
LSTM hidden size	256
LSTM layers	2
Source/target vocabulary size	25000
Teacher forcing	0.75
Window size	7

Table 1: The chosen values for the hyperparameters of both models. Note that "LSTM hidden size" and "LSTM layers" applies both to the encoder and decoder LSTMs. "Window size" only applies to "LA" since "S2S" does not use any attention mechanism.

6. RESULTS

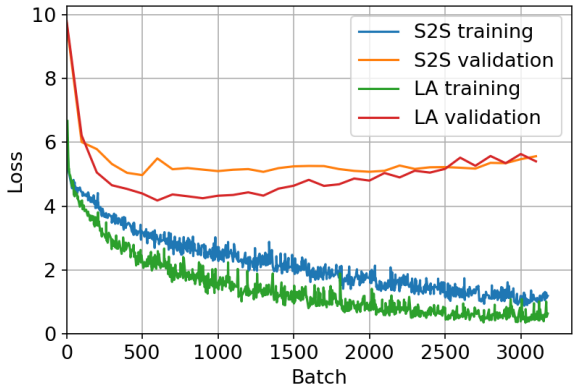
As previously stated, the BLEU of each model is evaluated on a part of the Multi30K dataset that is not used during training. The BLEU of each model can be seen in table 2. It is clear from table 2 that LA outperforms S2S greatly given that the BLEU of LA is almost twice as big as the BLEU of S2S. Considering that the difference between the two BLEU is this big, we think it is safe to say that local attention brings a significant improvement to the quality of the translation.

6.1. Learning Curves

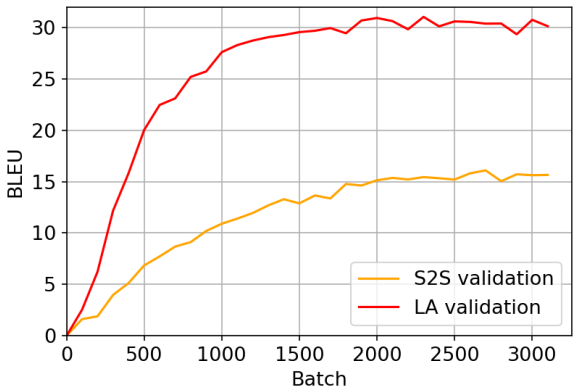
Figure 2a shows the loss as training progresses for both LA and S2S and figure 2b shows the same but for BLEU instead of the loss. We see in figure 2a that both of the models are overfitting to the training data. Interestingly, however, we see in figure 2b that the BLEU is not significantly affected by the overfitting in the first 3100 batches. It should be noted, that we in several

Model	**BLEU**
S2S	16.62
LA	30.37

Table 2: BLEU on the test set for each of the two models.



(a) Loss of S2S and LA.



(b) BLEU of S2S and LA.

Fig. 2: Learning curves for S2S and LA on the training- and validation sets.

experiments observed the BLEU drop below 30 for LA when training for more than 14 epochs. Therefore the BLEU is also sensitive to overfitting but it takes several batches for the BLEU to respond to the overfitting that was visibile in the loss many batches before. We have not fully investigated why this delay occurs but we hypothesize that it occurs because BLEU is less sensitive to word ordering whereas the loss is very sensitive to different word orderings in translation and target sentence. We have tried several techniques for reducing the amount of overfitting and while those techniques resulted in less overfitting, they also resulted in a lower BLEU. Since BLEU is the main metric that measures the quality of our model, we decided to not apply those regularization techniques.

6.2. Sample Translations

Table 3 shows two translations made by the LA model on examples from the test set. Both translations show how the model has learned to generally make fairly meaningful translations

Source	A long-haired , male musician is playing on a piano .
Target	Ein Musiker mit langen Haaren spielt Keyboard .
Translation	Ein langhaariger Ein Mann spielt Klavier .

Source	A muzzled greyhound dog wearing yellow and black is running on the track .
Target	Ein Windhund mit Maulkorb in gelb und schwarz läuft auf der Strecke .
Translation	Ein Windhund mit den gelb - schwarzen Trikot rennt auf der Rennstrecke .

Table 3: Two sample outputs from the LA model. "Source" is the English source sentence, "Target" is the German target sentence and "Translation" is the translation generated by the LA model.

even though it sometimes makes mistakes that a human would not make. For example it translates "male musician" into "Ein Mann" (meaning "a man") and it also fails to incorporate the word "muzzled" in the second German translation.

6.3. Attention Visualization

(Local) attention has a nice property that it is easy to visualize the alignments made by the attention mechanism. We see such a visualization in figure 3 where the alignments of the first source sentence and translation of table 3 is shown. It is interesting to observe how the alignments correspond well between several pairs of words. For instance there is a high alignment between the pairs "long-haired"/"langhaarier", "playing"/"spielt" and "piano"/"Klavier". We also observe that the model creates high alignments between "Ein" and "musician" twice. This might explain why the model erroneously translates "male musician" into "Ein Mann".

6.4. Comparison

The Multi30K dataset was used in the Conference on Machine Translation 2016 (WMT16) [11] challenge. Among the 18 participants, the highest BLEU was 35.1 and the average BLEU was 28.44. This places our model with a BLEU of 30.37 above the average but not in the top. Another interesting recent (2018) work is by Lample et al. [12] who use an unsupervised approach to machine translation to achieve a BLEU of 32.8 on the Multi30K dataset. We see from the above examples that our model achieves results close to state of the art models but it does not have a higher BLEU than those models.

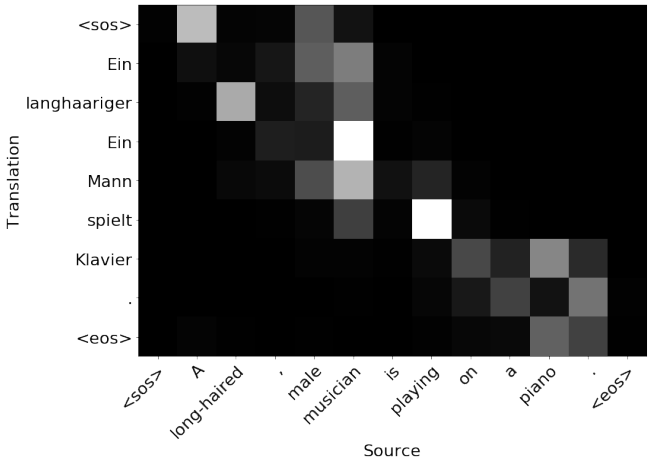


Fig. 3: Visualization of alignments in local attention when translating one word at a time from an English source sentence (x-axis) into a German translation (y-axis). Bright cells mean that the model has high attention to that part of the source sentence while dark cells mean that the model has low attention to that part of the source sentence.

7. CODE REPOSITORY

The project can be found on GitHub at the following URL: `https://github.com/gugundi/NeuralMachineTranslation`.

8. CONCLUSION

In this paper, we have implemented two models for neural machine translation (NMT): One model uses a local attention mechanism (LA) and the other uses a sequence-to-sequence approach without any attention mechanism (S2S). The models are trained and evaluated on separate parts of the Multi30K dataset consisting of around 30,000 English and German sentence pairs. The S2S model achieves a BLEU of 16.62 while the LA model achieves a BLEU of 30.37. Based on this, we conclude that local attention significantly improves the quality of the translation. The BLEU of 30.37 brings the LA model close to state of the art NMT models on the Multi30K dataset.

9. REFERENCES

[1] Todd Ward Kishore Papineni, Salim Roukos and Wei-Jing Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002, pp. 311–318.

[2] Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia, "Multi30k: Multilingual english-german image descriptions," *CoRR*, vol. abs/1605.00459, 2016.

[3] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

[4] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[6] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning, "Effective approaches to attention-based neural machine translation," *CoRR*, vol. abs/1508.04025, 2015.

[7] Jurgen Schmidhuber Sepp Hochreiter, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[9] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[10] David Zipser Ronald J. Williams, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, 1989.

[11] Khalil Sima'an Desmond Elliott Lucia Specia, Stella Frank, "A shared task on multimodal machine translation and crosslingual image description," .

[12] Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato, "Unsupervised machine translation using monolingual corpora only," *CoRR*, vol. abs/1711.00043, 2017.