Technical University of Denmark

02456

Deep Learning

Neural Machine Translation

Authors:
Christoffer Øhrstrøm
Gabriel K. Broholm

Study numbers:
s182827
s144454

November 1, 2018

Contents

1 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 (Kishore Papineni and Zhu, 2002) which is an automated evaluation method for machine translation that also works independently of source and target language. The experiments on the model are carried out on the IWLST 2017 evaluation campaign dataset (M. Cettolo and Federmann, 2017) but the model is evaluated on the larger Multi30K dataset (Elliott et al., 2016).

This paper will start in section 2 by referencing the related works by giving a brief outline of the different NMT implementations. In section 3 our model for this project is described, followed in section 4 by the experimental setup of the implementation.

2 Related Works

With the introduction of Long Short Term Memory (LSTM) 1997, it solved some of the issues that limited the previously existing recurrent neural network algorithms (Sepp Hochreiter, 1997). It solved those problems while only increasing the computational complexity - in terms of time and number of weights - by a constant factor and proved to train much faster by making the gradient less prone to vanish.

In 2002 the BLEU evaluation method was introduced and it became the standard evaluation method of NMT models (Kishore Papineni and Zhu, 2002). For the last decade different approaches to NMT have been introduced with different results and areas of focus. In September 2013 a state of the art language model was developed (Mikolov et al., 2013). Language models form the basis of many

natural language processing applications, including machine translation. A year later in September 2014 a new model using a encoder-decoder structure to process and recreate sequences of words, achieved a BLEU score of 33.3 (Cho et al., 2014). Just 3 months later a team from Google managed to achieve a BLEU score of 36.5, also using a encoder-decoder structure (Sutskever et al., 2014). Two years later Google again beat their own record with a BLEU score of 38.95 (Wu et al., 2016). Once more in December 2017 Google delivered state of the art results with a new transformer model, dispensing the encoder-decoder structure entirely on behalf of the attention mechanism (Vaswani et al., 2017).

In May 2016 the Multi30K dataset was released. This dataset is multilingual and contains translations from English to German image descriptions (Elliott et al., 2016). Since then multiple multilingual datasets have been released such as the IWSLT English to French dataset used in (Ngoc-Quan Pham, 2017). This project will be using these two datasets to make experiments, train and test the performance of our neural network. The IWSLT dataset does not require as much computational power as the Multi30K, making it ideal for experimenting and testing different setups. Once an ideal setup is found, the Multi30K will be used for evaluating the BLEU score.

3 Model

Our model seeks to generate the most likely target sentence (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 (Sutskever et al., 2014) with an attention mechanism (Dzmitry Bahdanau, 2016). This works by letting an encoder take an input sentence and transform it into a hidden state which is used by the attention mechanism to feed an input to the decoder which then generates a target sentence.

3.1 Encoder and Decoder

Both the encoder and decoder are recurrent neural networks (RNN). Specifically we use LSTM (Sepp Hochreiter, 1997) to make the RNNs more resilient to long sequences. That is we use a LSTM architecture 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 word probabilities in the target sentence. Similar to (Dzmitry Bahdanau, 2016), we let the encoder be a bidirectional RNN to make it capable of remembering not only previous seen words but also the following words of the input sentence. Historically many models (Sutskever et al., 2014, Cho et al., 2014) using this encoder-decoder architecture generates a single fixed sized context vector, c, as the output of the encoder which is then given as input to the decoder. While we will not do this, it is useful to briefly explain this architecture to understand our final architecture as seen in section 3.2.

Let $x = (x_1, x_2, ..., x_{T_x})$ be the input sentence where each x_i is a one-of-V_x encoding for a input vocabulary of V_x and let V_y be the target language vocabulary size. The decoder will then generate a target sentence $y = (y_1, y_2, ..., y_{T_y})$ where

each y_j is a vector of length V_y. While generating the jth target word, the decoder will compute

$$p(y_j|\{y_{j-1}, ..., y_1\}, x) = g(y_{j-1}, s_j, c) \tag{1}$$

where g is some non-linear function that computes a probability distribution over the vocabulary of the target language and s_j is the hidden state of the decoder at time j. The probability of the target sentence y is therefore

$$p(y|x) = \prod_{j=1}^{T_y} p(y_j|\{y_{j-1}, ..., y_1\}, x) = \prod_{j=1}^{T_y} g(y_{j-1}, s_j, c) \tag{2}$$

for a target sentence of length T_y.

3.2 Attention Mechanism

In (Dzmitry Bahdanau, 2016) the authors argue that the fixed size context vector in the encoder-decoder setup creates a bottleneck for translation performance. As a solution, they propose an attention mechanism that enables the decoder to focus (pay attention) to specific parts of the input sentence. This means that the jth output word becomes dependent on a vector c_j rather than all output words depending on the same c as in equation 1 and 2. Introducing this change, changes equation 1 into

$$p(y_j|\{y_{j-1}, ..., y_1\}, x) = g(y_{j-1}, s_j, c_j) \tag{3}$$

and equation 2 into

$$p(y|x) = \prod_{j=1}^{T_y} p(y_j|\{y_{j-1}, ..., y_1\}, x) = \prod_{j=1}^{T_y} g(y_{j-1}, s_j, c_j) \tag{4}$$

The encoder produces T_x annotations $(h_1, ..., h_{T_x})$ for an input sentence of length T_x and each h_i is a concatenation of the forward states and backward states of the bidirectional RNN encoder. An annotation h_i therefore encodes the entire input sentence and at the same time h_i is focused around the ith input word due to the nature of RNNs to focus on recent inputs. This makes the annotations useful in computing which parts of an input sentence to pay attention to. Each c_j is computed as an expectation over annotations which we now will show how to compute. Let s_j be the jth hidden state of the decoder RNN and let the alignment model a be a feed forward network. We then compute the energy from the ith input word to the jth output word:

$$e_{ij} = a(s_{j-1}, h_i) \tag{5}$$

Note that an important property of a is that it is trained together with the rest of the model. The energy is then used to compute α_{ij} as a probability distribution over how likely the ith input word matches the jth output word.

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{kj})} \tag{6}$$

α_{ij} can then be used to compute the expectation over annotations which gives us the desired context c_j.

$$c_j = \sum_{k=1}^{T_x} \alpha_{kj} h_k \tag{7}$$

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

4 Experimental Setup

4.1 Framework

The framework used for implementing the neural network will be the Python 3.6 PyTorch framework. The reasoning behind this is that it provides an easy and intuitive approach to modelling neural networks and comes with a comprehensive documentation. It allows for quick creation and editing of the network.

4.2 Hardware specifications

Experimentation and testing will be performed using the DTU High Performance Computing cluster[1]. When experimenting with the neural network model, the TitanX/Tesla cluster will be used, although the hardware is a shared resource among DTU students. For the actual testing on evaluation of the BLEU score, the more powerful Volta node cluster will be used.

4.3 Hyper parameters

The hyper parameters of the network can be seen in the table 1:

Hyper parameters	Values/Type
Learning rate	TBD
Regularization	TBD
Optimizer	TBD
Weight decay	TBD
Dropout	TBD

Table 1: An overview of the values or type of the different hyper parameters.

References

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Dzmitry Bahdanau, Kyunghyun Cho, Y. B. (2016). Neural machine translation by jointly learning to align and translate.

[1] `https://www.hpc.dtu.dk/?page_id=2129`

Elliott, D., Frank, S., Sima'an, K., and Specia, L. (2016). Multi30k: Multilingual english-german image descriptions. *CoRR*, abs/1605.00459.

Kishore Papineni, Salim Roukos, T. W. and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

M. Cettolo, M. Federico, L. B. J. N.-S. S. K. S. K. Y. and Federmann, C. (2017). Overview of the iwslt 2017 evaluation campaign, in proceedings of the 14th international workshop on spoken language translation (iwslt), tokyo, japan.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Ngoc-Quan Pham, Matthias Sperber, E. S. T.-L. H. J. N. A. W. (2017). Kit's multilingual neural machine translation systems for iwslt 2017.

Sepp Hochreiter, J. S. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

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., Kaiser, L., 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. *CoRR*, abs/1609.08144.