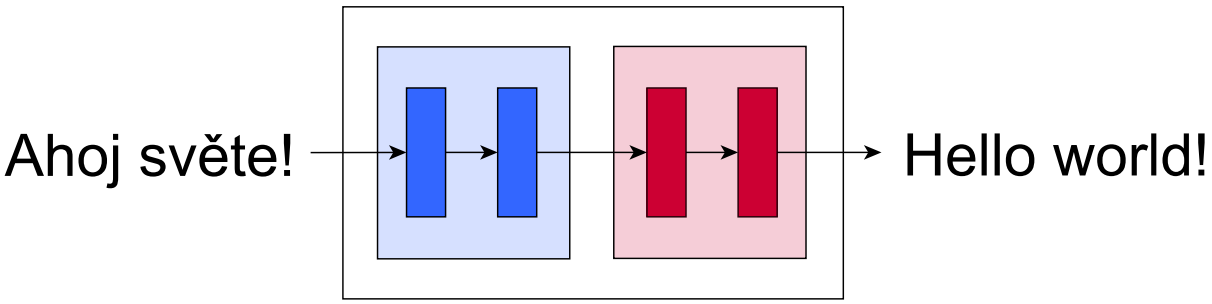
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2018
http://excel.fit.vutbr.cz

Machine Translation Using Artificial Neural Networks

Jonáš Holcner*



Abstract
The aim of this work is to develop a neural machine translation system (NMT). That is a machine translation system based on neural networks. Specifically, the system is based on the encoder-decoder architecture, created with recurrent neural networks enabling sequence to sequence translation.
The system is build with libraries Keras and Tensorflow and is tested against Moses statistical machine translation tool.
This work does not bring some concrete model with new state of the art results but shows some insight into the topic as well as provides an open source python library that can interested reader use to easily conduct his own experiments.

Keywords: neural machine translation — NMT — recurrent neural networks — RNN — LSTM — encoder-decoder architecture — sequence to sequence — seq2seq — keras — moses — BLEU

Supplementary Material: Downloadable Library

*xholcn01@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

In the recent years, there is a significant increase in usage of machine learning and artificial intelligence. This is because only lately, the capacity and performance of computers caught up with the available amount of data that is being produced every day as to build and train large enough neural networks. Now days, neural networks are widely capable of recognizing images, transcribing spoken language and most interestingly for this paper, they are quite capable in translating sequences from one language to another.

The biggest advantage of modern NMT approach is that it does not have some of the problems the traditional machine translation systems had. Instead of being composed of many different complex parts, NMT has the ability to learn and translate directly in end-to-end fashion.

Goal of this work is to develop and try out such system and provide an out of the box usable library. Solution proposed in this paper make use of the encoder-decoder architecture. Each of these two components is one recurrent neural network together capable of directly translating whole sequences from one language to another.

The result is python package *nmt* built with Keras and Tensorflow. With this package were conducted experiments, evaluated with the standard BLEU score. Results were compared with the system produced by the Moses [1] statistical machine tool.

2. Previous Works

First idea of recurrent neural networks comes from the nineties [2]. The vanilla RNN, however, had a problem with long term dependencies because of the vanishing and exploding gradient [3].

Thus came improved variants of the RNN – long short term memory (LSTM) [4, 5] and its simpler version, gated recurrent unit (GRU) [6]. These units have a memory, that stores and changes information in it over time, enabling the network to remember long term dependencies.

Works [7, 8, 9] shows that good performing language models are possible to build with recurrent neural networks. This lays foundation for the neural machine translation as language models are the vital part. The advantage of neural language model is that it learns embeddings in a continuous space for the words, which provides the model with more context it can learn from. Different variants of learning the word embbeddings are shown here [10, 11, 12, 13]. Pretrained word embeddings, for example on some very large data set, can be used to boost performance of a NMT system, which would have to otherwise learn those embeddings by itself.

Encoder-decoder architecture was proposed in [14] and was used for rescoring hypotheses produced by a phrase-based system with successful improvement. Paper [15] then shows how to use encoder-decoder architecture for direct sequence to sequence translation and comes with the best results at the time. Furthermore, it was found that the importance of reversing order of the words in all source sentences (reverse encoder), improves models performance, by introducing short term dependencies between the source and the target sentence.

Upon this builds work [16] which shows even better results with bi-directional encoder which is encoder that process in both normal and reverse order. What is even more important, they address the problem of encoder-decoder approach, where the meaning of the translated sentence is captured in a fixed-length vector and that can be problematic for translating long sentences. The proposed remedy is so called *attention* mechanism which lets the model, at the time of decoding, look at the most important words from the source sentence for the currently translated word, resulting in even better performance.

As translation is an open-vocabulary problem, the NMT systems have to somehow handle the words not present at the time of the training. This was typically done by using out-of-vocabulary tokens and by using very large vocabularies, which causes the models to be very memory and performance demanding. Solution for this can be sub-word units [17, 18], that are shown to be more efficient, help with rare and unknown words and improve the results.

Current state-of-the-art results are published by Google [19, 20], which uses all of the techniques described, showing that they can be successfully applied on large production data sets.

Another thing Google shows is, that with no changes to the model architecture, one model can be used to learn to translate between many languages [21], even to produce translations between languages that it was not explicitly trained on (zero-shot translations).

3. Seq2seq translation with encoder-decoder architecture

Sequence to sequence (**seq2seq**) means that the process of translating takes one whole sentence at the input, in a source language, and its meaning translates into sequence in a target language. This can be better in contrast to other methods like word-based translation or phrase-based translation as in seq2seq the whole meaning of the source sequence is translated at once and can keep the precise meaning. Seq2seq is modeling probability of a target sentence based on source sentence as shown in figure 1.

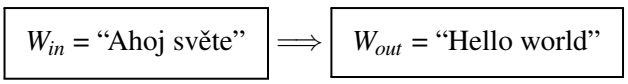


Figure 1. Seq2seq models a probability $P(W_{out}|W_{in})$. Meaning that the model learns to predict sentence W_{out} based on a sentence W_{in}.

Neural network architecture, used for this kind of modeling, is **encoder-decoder**. Both encoder and decoder are recurrent neural network (figure 2), usually one or more layers of LSTM or GRU.

Encoder takes embeddings of the input tokens (words or subwords) and processes the input sequence in source language into fixed-length vector, so called “thought” vector – it captures the essence of the given sentence.

Decoders hidden state is initialized with this output from encoder. The decoding process is started with start token, which is given to decoder as first input. Decoder then generates sequence in target language until

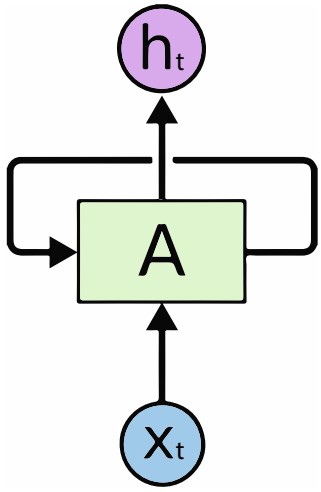


Figure 2. RNN are neural networks with loops. They process sequence of data and output h_t from each time step t is fed as input to time step $t+1$. Image taken from [22].

it reaches end token that tells it to stop. There is a difference between time when model is being trained and between time when it is used to predict sequences. In the training time, decoder is fed the correct, expected output. This is called "teacher forcing" [23]. In the inference time, decoder is fed its own output each time step. Figure 3 shows the encoder-decoder architecture.

Equations of the encoder-decoder architecture:

$$m_t^{(f)} = M_{f_t}^{(f)} \tag{1}$$

$$h_t^f = \begin{cases} RNN^{(f)}(m_t^{(f)}, h_{t-1}^{(f)}) & \text{if t} \geq 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

$$m_t^{(e)} = M_{e_{t-1}}^{(e)} \tag{3}$$

$$h_t^e = \begin{cases} RNN^{(e)}(m_t^{(e)}, h_{t-1}^{(e)}) & \text{if t} \geq 1, \\ h_{|F|}^f & \text{otherwise.} \end{cases} \tag{4}$$

$$p_t^{(e)} = \text{softmax}(W_{hs} h_t^{(e)} + b_s) \tag{5}$$

- $RNN^(f)$ - encoder, $RNN^(e)$ - decoder
- $M_{f_t}^{(f)}$ - embedding for encoder input in time t
- $M_{e_t}^{(e)}$ - embedding for decoder input in time t
- $h_t^{(f)}$ - encoder hidden state in time t
- $h_t^{(e)}$ - encoder hidden state in time t
- W_{hs} - weights
- b_s - bias

Embedding is looked up for every word in time t (eq. 1). Then the hidden state of encoder is computed (eq. 2). After processing the whole input sequence, there should be enough context stored for decoder initialisation. Then, as for encoder, embedding is looked up for decoder input (eq. 3). Only now the word is from time $t-1$ as decoder generates new word based on the last one. In the time t_0 decoder is fed starting symbol $\langle s \rangle$. Eq. 4 computes hidden state of decoder and shows that in the time t_0, decoder is initialised with encoder final state. Final output probability is resolved with $softmax$ function (eq. 5).

For a deeper overview of NMT based systems, I would point the reader to [24].

4. Implementation of the NMT system

datasets preprocessing bucketing fit generator? bidirectional encoder subwords - BPE beam search shuffling main.py repo

5. Experiments and evaluation

6. Conclusions

[Paper Summary] What was the paper about, then? What the reader needs to remember about it?

[Highlights of Results] Exact numbers. Remind the reader that the paper matters.

[Paper Contributions] What is the original contribution of this work? Two or three thoughts that one should definitely take home.

[Future Work] How can other researchers / developers make use of the results of this work? Do you have further plans with this work? Or anybody else?

Acknowledgements

I would like to thank my supervisor Ing. Igor Szőke, Ph.D. for his help.

References

[1] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[2] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[3] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, March 1994.

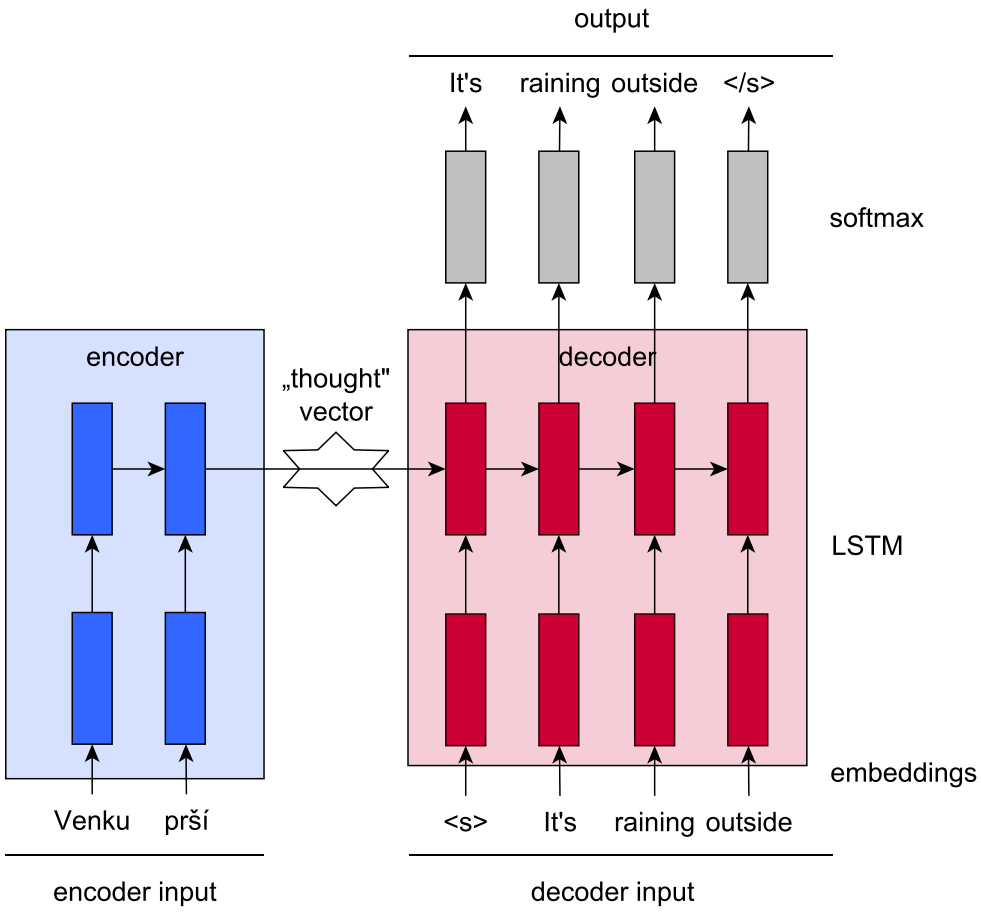


Figure 3. Encoder-decoder architecture shown on translation from Czech sentence to its English equivalent. Encoder processes embeddings of input sequence and produces fixed-length "thought" vector. This vector is used as initial state of decoder, it tells it from what context should it produce output in target language. Prediction is started with the $\langle s \rangle$ starting token. Then the decoder is fed either correct output tokens during training time or its own output, from time $t-1$, during inference time, until it generates the ending $\langle /s \rangle$ token.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[5] Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with lstm. *Neural Comput.*, 12(10):2451–2471, October 2000.

[6] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[7] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[8] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252. Society for Artificial Intelligence and Statistics, 2005.

[9] Tomáš Mikolov. *Statistické jazykové modely založené na neuronových sítích*. PhD thesis, Vysoké učení technické v Brně, Fakulta informačních technologií, 2012.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[11] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751. Association for Computational Linguistics, 2013.

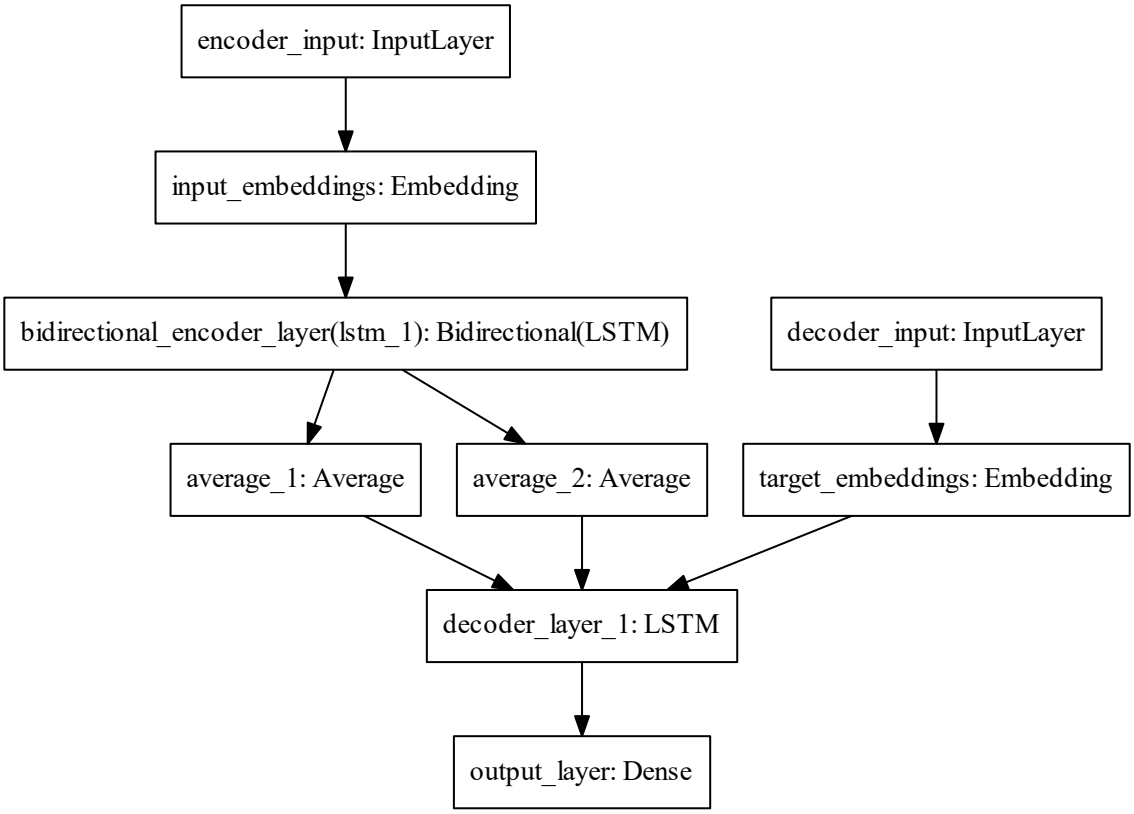


Figure 4. Layers of model in keras

[12] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[13] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016.

[14] 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*, abs/1406.1078, 2014.

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

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

[17] Tomas Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocký. Subword language modeling with neural networks. In *Subword Language Modeling with Neural Networks*, 2011.

[18] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015.

[19] 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, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar,

Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[21] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR*, abs/1611.04558, 2016.

[22] Christopher Olah. Understanding lstm networks, 2015. [Online; navštíveno 3.12.2017].

[23] Anirudh Goyal, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. pages 4601–4609. 2016.

[24] Graham Neubig. Neural machine translation and sequence-to-sequence models: A tutorial. *CoRR*, abs/1703.01619, 2017.