NEURAL MACHINE TRANSLATION FOR GERMAN-ENGLISH TRANSLATION

Thomas Nilsson, tnni@dtu.dk

Technical University of Denmark
DTU Compute
Building 324, 2800 Kongens Lyngby,
Source code: *github.com/thomasnilsson/02456-handin-s144470*

ABSTRACT

Neural Machine translation (NMT) is behind recent long strides within automation of translation. In this paper we explore using two deep learning models for NMT for translating German sentences to English. The dataset used was the Multi30k [1] which contains 29,000 training examples. The architectures used were the RNN Encoder-Decoder published in 2014 [2] and the Transformer published in 2017 [3], which produced BLEU scores of 10.56 and 29.28 respectively on the test dataset, after 50 epochs. The Transformer is clearly the superior model, but the BLEU score is still somewhat lower than what is claimed by other research, using the same Transformer model on the same dataset.

Index Terms— Deep Learning, NLP, Transformer, RNN, Machine Translation

1. INTRODUCTION

Machine translation is the task of generating an output sentence in a target language, English for example, given an input sentence in a source language such as German. A naive machine translation algorithm will rely on simple substitution of words in the source sentence to create the target sentence language, however, this approach has several problems related to how phrases need to be considered as a whole to find the most correct translation and that the two sentences will likely not have the same number of words in them. Such an example is demonstrated by the following German-English sentence pair:

```
Maschinenübersetzung is toll

Machine translation is great
```

This happens due to how the German language combines words, in contrast to the English language which keeps the words 'Machine' and 'translation' separate.

1.1. Neural Machine Translation

Neural machine translation (NMT) is behind big recent leaps in Machine Translation. NMT takes a deep learning approach to aligning sentences and extracting dependencies between them and does not rely on simpler concepts such as alignment or statistical models. Specifically, Sequence-to-Sequence (seq2seq) models are employed which are deep learning models that generate an output sequence given an input sequence, i.e. given a German sequence of words, the equivalent English sequence is produced, and such a seq2seq model is a translation model.

This paper will explore the use of two recent state of the art models within Neural Machine Translation to translate German sentences from the Multi30k [1] to their equivalent English sentences.

1.2. BLEU Scoring for Translation

For evaluating Machine Translation models, the BLEU [4] metric is used which compares a candidate sentence, i.e. the predicted sentence to the reference sentence, i.e. the ground truth sentence, by counting the N-gram overlap, up to $N = 4$. There are a few ways the BLEU score can be calculated through smoothing to avoid division-by-zero errors for sentences shorter than 4. However, the default BLEU score is simply a weighted product of the number of N-gram overlaps as well as a brevity penalty applied to the candidate sentence, the latter punishing translations that are shorter than their reference, since otherwise producing a very short translation would cheat the system and get a high score.

$$BLEU = \min\Big(1, \frac{len(candidate)}{len(reference)}\Big) \cdot \Big(\prod_{n=1}^{4} P_n\Big)^{1/4}$$

2. DATASET

The Multi30k dataset [1] contains $29,000$ sentence pairs in the training set, and $1,000$ sentence pairs in both the

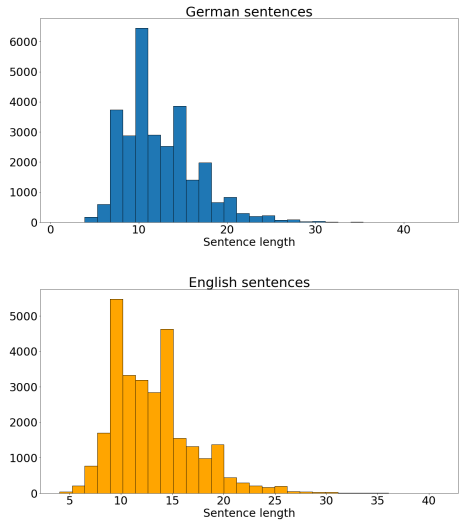

Fig. 1. Sentence length distributions for the German- and English sentences. The dataset contains sentences of many different lengths, which may impact the BLEU scoring of the results since longer sentences are very hard to translate word for word compared to a reference.

validation- and test set. The sentence length distribution for both German- and English sentences is displayed in 1.

German combines adjectives and nouns into a new noun as mentioned previously, leading to a larger vocabulary due to simply having more words to cover the same sentences, which also implies that some German sentences will be shorter than their English counterparts.

2.1. Word Embeddings and Pre-processing

In order to relate the semantic meaning of single words found in the corpus, the machine learning models make mathematical representations in the form of vector-space embeddings. This can be done in many ways such as with pre-trained GloVe embeddings [5] which have been found by building a co-occurrence matrix on a large text corpus that finds words which often occur in the same context. However, no pre-trained word embeddings were used in this project. The dataset had limited pre-processing applied to it, which includes excluding words which occur fewer than 2 times in their language corpus, and replacing those words with the `<UNK>` token. The main reason for doing this being that the embedding matrix for a language contains a row for each word in the language's vocabulary, which means the matrix can grow incredibly large if very rare words and misspellings are used as well. However the downside is that named entities, such as names of companies or people will all be mapped to the same word, i.e. the `<UNK>` token. More sophisticated embedding techniques include character-level embeddings [6] which builds embeddings based on parts of words and characters rather than whole words, thus solving the problem of out-of-vocabulary words. This approach was however not considered for this project. The last part of the pre-processing is appending sentences with the `<SOS>` (start-of-sentence) and `<EOS>` (end-of-sentence) tokens, to indicate where the sentence begins- and where it ends.

2.2. Batching

For arranging the dataset to be run through the models a `Dataset` sub-class was written for `PyTorch` which allows the dataset to be split into batches. A *batch* is a collection of input- and output sentences, with some batch size. For the batch to be run through the model it is required to be a matrix, which implies that all the input-sentences have the same length and vice versa for the output sentences.

Batching allows the model to train on multiple examples at once, by flattening the two batch-matrices of inputs and outputs into two vectors, generating a vector from the given input and then calculating a loss from the prediction and the target vector. This means that the back-propagation is run for each batch, rather than for each example which in turn means fewer parameter updates which mean less compute time per example. In essence, the larger batch size is better, however, the batch size is limited by the available memory on the hardware the program runs on, and more specifically the RAM. A too-large batch size loaded into RAM causes the program to crash, and therefore the batch size is specific to the dataset and the model.

Converting a set of variable length sentences into a matrix is achieved through *padding*, which is the process of appending the `<PAD>` token to sentences shorter than the maximum sentence length within the current batch. Optimally, as little padding as possible should be used which is desired due to wanting the smallest possible input and output for computation reasons. This is achieved by arranging the data such that the inputs- and outputs in a batch are chosen based on their length, which minimizes the amount of padding used for the batch, i.e. no batch should contain both long sentences and short sentences.

3. SEQ2SEQ MODELS

3.1. RNN Encoder-Decoder

For a long time, the Encoder-Decoder architecture proposed Cho et al. [2] in 2014 was the gold standard within Seq2Seq tasks such as Neural Machine Translation. The model uses two separate Recurrent Neural Networks (RNNs), one being trained to encode a variable-length sentence from the source language into a fixed-length vector representation, and the other network trained to decode this intermediate state into tokens in the target language. The architecture is depicted in Figure 2.

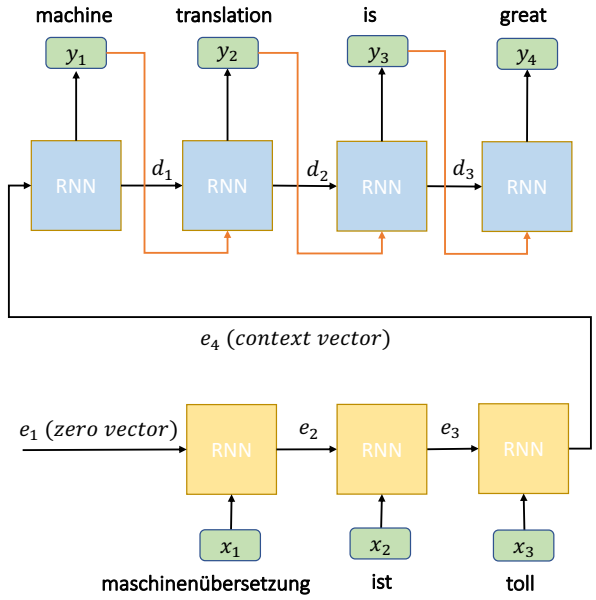

Fig. 2. The RNN Encoder-Decoder model architecture for a German-English sentence pair that contains a different number of tokens. For decoding the next token, the previous decoder output is used, this is illustrated with the orange arrows. Teacher forcing may alternatively be used, where the target token at a given time-step is fed to the decoder.

The upside of this architecture is that it allows for a variable-length input, whereas traditional neural networks enforce a strict input size. However, the downside is that the input sentence is compressed into a fixed-size representation regardless of whether the input is very long or just a single token. This is referred to as the information bottleneck problem and is addressed by Bahdanau, Cho and Bengio the same year, [7] where the *Attention* mechanism is employed to alleviate this problem of capturing long-range dependencies. The attention mechanism is an additional score calculated for each decoding step to determine which encoding step is the most relevant. In its basic form just a dot product between the current decoder hidden state d_t and every other encoder hidden state $[e_1 ... e_n]$ for an input of size n. The idea is that a large, positive dot product means the two vectors are aligned and thus are semantically related, which gives an indication of which parts of the two sentences are semantically related.

3.2. The Transformer

In 2017 it was proposed by Google Brain and Google Research in 2017 [3] to scrap the recurrence, and instead let attention solve perform the translation alone. The Transformer model can be heavily parallelized with multi-threading and GPU training since there is no sequential bottleneck like in RNNs, where the sentences are unpacked iteratively. It is, therefore, a much faster model to train and will reach the performance of the RNN-based models faster. The core mechanism of the Transformer is the self-attention, which calculates the attention score between each element of a given sequence, i.e. the input or the output sentence. This attention mechanism is employed in parallel, and in total 8 attention-heads are used in the original paper for the Transformer. The input to the Transformer is read all at once, in parallel which means the sense of order in a sentence is lost and must be provided explicitly. To solve this issue, a positional encoding is used which is a vector representing an embedded position for each word which sin- and cos functions are used. Concretely the positional embedding is calculated using the position of a word pos and the dimension of the embedding i as

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/dim_{model}}}\right)$$
$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/dim_{model}}}\right)$$

are used. The trigonometric functions are useful since they always provide a unique positional encoding between -1 and 1 for each word in the sentence.

Firstly an attention product is calculated for the source sentence with itself, and the same is done for the target sentence. For the output sentence, masked-attention is used to prevent revealing future words to the model, at the current 'time step'. These self-attention products identify long-range dependencies within the two sentences and avoid the problem of an information bottleneck that the previous model suffered from. After this, the attention product is calculated between the source and target sentence. The other major upside of this model is that it allows for heavy parallelization since the sentences are not iterated through one token at a time, but rather all at once, making the model much faster - both in terms of training and adjusting parameters but also in terms of generating sentences.

The hyper-parameters for the Transformer were chosen according to the Machine Translation Model proposed in the original paper [3] which has over 54 M parameters. Due to the model over-fitting the dataset as is discussed in Chapter 4, likely due to the high parameter count, a smaller version of the Transformer was created by lowering the feed-forward dimension, denoted dim_{FF} from 2048 to 512. This model will be denoted *Transformer512* and the original implementation with size 2048 will be denoted *Transformer2048*. The concrete parameters for the Transformer models can be found in Appendix A, in Table A.2.

3.3. Training

For calculating the error performed by the model's prediction $\mathbf{p}$ and the target sentence $\mathbf{y}$ the Cross-Entropy Loss function was used, which compares every word in the predicted sentence to every word in the target sentence. The two sentences are both guaranteed to have the same length due to

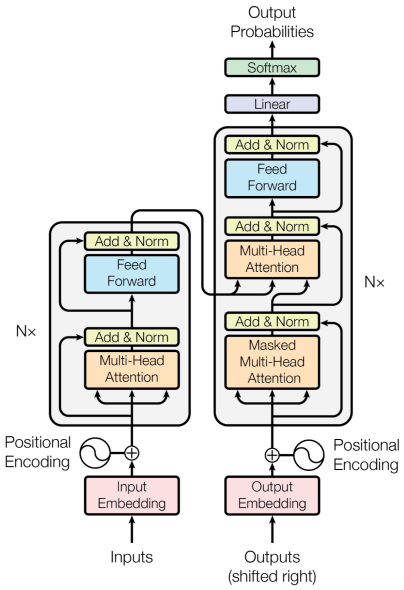

Fig. 3. The Transformer architecture where encoding and decoding happens all at once. The Transformer uses several encoder-decoder layers which is marked by the Nx label, N meaning the number of stacked layers. Figure from Google Research [3].

padding, however, Masked Cross-Entropy loss is used to deal with the issue, which excludes any indices i for which $t_i = 0$ i.e. the <PAD> token. Given a sentence S containing words $w_1, ..., w_N$ the loss is calculated as.

$$loss = -\sum_{i=1}^{N} t_i \cdot \log(p_i) \quad \forall i \in [1, N] \quad t_i \neq 0$$

For the RNN Encoder-Decoder the Adam Optimizer with default parameters was used and for the Transformer, Adam was also used, but with the parameters specified in A.2 in Appendix A and the models were each trained for 50 epochs.

4. RESULTS AND DISCUSSION

Model	Params	Test Loss	Test BLEU
RNN	7.24 M	3.48	10.56
Transformer-512	35.3 M	1.70	27.57
Transformer-2048	54.2 M	1.76	29.28

Table 1. BLEU scores for each model on the test-set as well as their respective parameter counts.

The best BLEU score, and by proxy the lowest loss is reached using the *Transformer2048* model with a parameter count of 54.2 M. This model over-fits after just 10 epochs

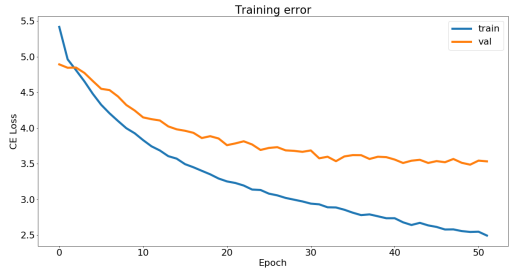

Fig. 4. Validation loss for the RNN Encoder-Decoder over 50 epochs. The model keeps improving however the validation loss is still quite high even after 50 epochs.

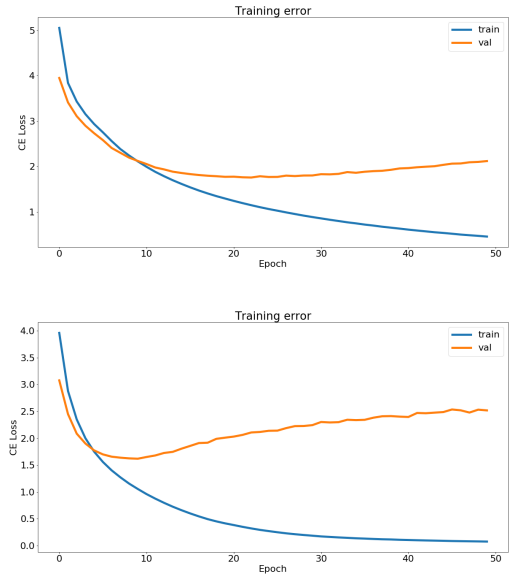

Fig. 5. Validation loss for the *Transformer512* (top) and *Transformer2048* (bottom). Both models over-fit the dataset eventually: For the *Transformer512* it happens after 20 epochs and the *Transformer2048* does it after just 10 epochs. This indicates the models are likely too complex wrt. the number of parameters for the dataset.

(Figure **??**) in contrast to the RNN Encoder-Decoder which keeps on minimizing its validation loss steadily with each epoch (Figure 4). The validation loss of the *Transformer2048* is however under half of the RNN Encoder-Decoder meaning it is by far superior.

For benchmarking the *Transformer2048* the following implementation from *The Distributed (Deep) Machine Learning Community* [1] is looked upon, which achieves 35.41 BLEU with a Transformer model on the Multi30k dataset, although the exact hyper-parameters are not disclosed at it is assumed the parameters are the same as the original Transformer paper i.e. the same parameters - however, a modified Adam optimizer is used which may explain how it can achieve a higher BLEU score.

4.1. Further Work

The dataset is likely too small to justify the large parameter count, and likewise, a smaller parameter count will probably limit how much the model can learn about the dataset, i.e. the solution is to acquire a larger dataset such as the WMT-14 German-English dataset [2] which consists of 4.5 million sentence pairs, and is the dataset used in the original Transformer paper [3].

5. REFERENCES

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

[2] 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.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[4] Kishore Papineni, Salim Roukos, Todd Ward, 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*, Philadelphia, Pennsylvania, USA, July 2002, pp. 311–318, Association for Computational Linguistics.

[5] Jeffrey Pennington, Richard Socher, and Christopher Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1532–1543, Association for Computational Linguistics.

[6] Lyan Verwimp, Joris Pelemans, Hugo Van hamme, and Patrick Wambacq, "Character-word LSTM language models," *CoRR*, vol. abs/1704.02813, 2017.

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

[1] https://github.com/dmlc/dgl/tree/master/examples/pytorch/transformer

[2] https://nlp.stanford.edu/projects/nmt/

A. MODEL PARAMETERS

A.1. RNN Encoder-Decoder

Model	batch size	layers	$\dim_{hidden}$	$\dim_{embedding}$	dropout
RNN-Encoder-Decoder	256	2	256	256	0.5

Table 2. Parameters for the RNN Encoder-Decoder model.

A.2. The Transformer

Model	batch size	heads	layers	$\dim_{model}$	$\dim_{FF}$	dropout
Transformer-512	32	8	6	512	512	0.1
Transformer-2048	32	8	6	512	2048	0.1

Table 3. Parameters for the Transformer models.

Optimizer	**learning-rate**	**beta1**	**beta2**	**epsilon**
Adam	1e-4	0.9	0.98	1e-9

Table 4. Parameters for the Adam optimizer used for both Transformer models.

B. EXAMPLE TRANSLATIONS

Below, a subset of the German-English translations are shown. The translations were produced by the best-performing model, i.e. the *Transformer2048* model.

```
Original (de): arbeiter diskutieren neben den schienen .
Target (eng): workers workers are a discussion next the under .
Predicted (eng): construction workers having a discussion by the tracks .

Original (de): zwei jungen spielen gegeneinander fußball .
Target (eng): two boys are soccer with each other .
Predicted (eng): two boys play soccer against each other .

Original (de): zwei fußballmannschaften auf dem feld .
Target (eng): two soccer players are running the field .
Predicted (eng): two soccer teams are on the field .

Original (de): ein hellbrauner hund läuft bergauf .
Target (eng): a tan brown dog is running with some
Predicted (eng): a light brown dog is running up .

Original (de): leute bewundern ein kunstwerk .
Target (eng): people are making a piece piece art .
Predicted (eng): people are admiring a work of art .

Original (de): ein junge steht mit drei mädchen .
Target (eng): a boy is with three girls .
Predicted (eng): a boy stands with three girls .

Original (de): zwei männer sitzen in einem restaurant .
Target (eng): two men sitting in a restaurant .
Predicted (eng): two men sitting in a restaurant .

Original (de): zwei typen und eine frau lächeln .
Target (eng): two guys and one woman smiling .
Predicted (eng): two guys and a girl smiling .

Original (de): zwei kleine kinder auf dem sand .
Target (eng): two young children are sand the .
Predicted (eng): two young children are on sand .

Original (de): ein kleiner schwarzer hund springt über gatter
Target (eng): a small black dog jumps over a
Predicted (eng): a small black dog jumping over gates
```