Decoder Only Machine Translation

Christoph Schneider

Abstract

This work presents a novel approach to machine translation. The author implements a Transformer like architecture that is able to auto-regressively translate without using an encoder. Experiments show that this architecture does not reach the same performance as the Transformer even though its hyperparameters were tuned whereas the Transformers were not. Validation steps take longer and the architecture is more susceptible to suboptimal hyperparameter settings. The project is implemented as a fork of JoeyNMT [1].

1 Introduction

Encoder-Decoder models have been the de facto standard for machine translation and sequence to sequence learning in general, since they have been introduced by Sutskever et al. (2014). In this framework, the encoder maps the source sequence to a fixed length vector or a series of vectors. The decoder produces the target sequence one token at a time (auto-regressively). It is conditioned on its own output and the output of the encoder, often using an attention mechanism (Bahdanau et al., 2014). The aim of this term project is to explore a possible alternative to the encoder-decoder framework. Instead of letting an encoder produce a vector representation of the source sequence, we concatenate source and target sequence, seperated by a special separator token. A Transformer (Vaswani et al., 2017) decoder without encoder-decoder attention is trained on those sequences. During inference we pass the source sequence followed by the separator token to the decoder and let it auto-regressively produce the target sequence until it produces the end-of-sequence token.

[1]The implementation can be found at: https://github.com/HerrKrishna/joeynmt/tree/DecoderOnlyMT

2 Model

Our model is the decoder part of the Transformer without encoder-decoder attention since there is no encoder output to attend to. One could also say that we use the Transformer encoder as a decoder. The model converts the one-hot encoded input sequences $x = x_1, ..., x_m$ into f-dimensional embeddings $e = e_1, ..., e_m$ where w_j is a row in an embedding matrix $E \in \mathbb{R}^{V*d}$. Relative positional encodings $p = p_1, ..., p_m$ are added to the embeddings to give the model a sense of the position of input words in the sequence. The so obtained vector represention of the input z are processed by a series of layers, each of which consists of a multi-headed self-attention mechanism (Vaswani et al., 2017), layer normalization (Ba et al., 2016) and a feed-forward neural-network. The self-attention and feed forward sublayers are surrounded by residual connections (He et al., 2015). Given z the model produces a series of output tokens $y = y_1, ..., y_n$ one step at a time, at each time appending its last output token to the input sequence until it outputs the end-of-sentence token.

3 Decoder Only MT

Our model is simply the Transformer encoder used as a decoder. However to achieve machine translation without using an encoder, we have to introduce a few important alterations to how we usually use Transformers.

3.1 Inputs

Since our model doesn't use an encoder to produce a vector representation of the source sequence on which the decoder can be conditioned, we need another way of passing information about the source sequence to the decoder. To achieve that, we concatenate the source and the target sequence, sepa-

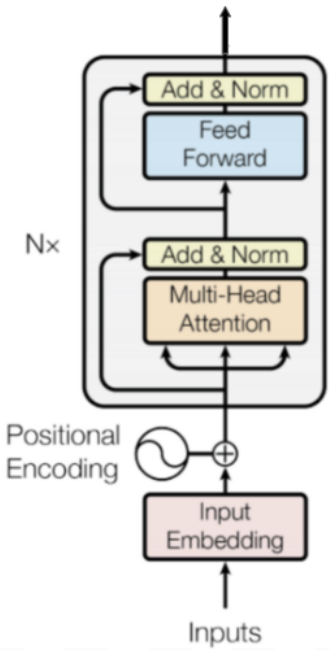

Figure 1: The first of many layers of the model

rated by a separator token.

- Source: I like cake
- Target: Ich mag Kuchen
- Input: <BOS> I like cake <SEP> Ich mag Kuchen
- Output: I like cake <SEP> Ich mag Kuchen <EOS>

We use a shared vocabulary between source and target in order to be able to concatenate as shown.

3.2 Masking

In the conventional Transformer decoder, during training with teacher-forcing, a look-ahead mask is used to prevent the self-attention mechanism from attending to future tokens since this would not be possible during inference due to the auto regressive nature of the model. In this work however, during inference the model starts to decode with the source sequence as an input. This means that during training, we can allow the model to freely attend in the source part of the input. In order to do this we unmask all the tokens in source part of the input.

3.3 Batching

To obtain minibatches, we sort the input sequences by length and put sequences of similar length into a minibatch. By using this strategy we accelerate training by reducing the amount of padding needed in each minibatch.
If we want to use minibatches at test time, we need to adjust the way we choose sequences for a batch. During inference all the sequences in a batch need to be of the same length and we can not use padding. This is not a problem in normal Transformers since every sequence in a minibatch starts out as only the <BOS> token. In our Decoder Only model we start inference with the source sequences which can be of different length. To prevent this, we sort the input sequences by the length of their source part and make sure that all the sequences in a batch have equally long source parts.

4 Dataset

The Dataset that was used in this experiment is IWSLT14 Dataset with German and English sentence pairs. The dataset was preprocessed using Byte Pair Encodings. Table 2 shows a summary of the dataset statistics.

	sentence pairs
train	156 206
dev	7 245
test	6 750

Table 1: Number of sentence pairs per split.

5 Experiments

We compare Decoder Only machine translation to regular machine translation. First we train a conventional Transformer model to obtain a baseline result. We don't do do any hyperparameter tuning for this system.
Then we train a Decoder Only model and tune its hyperparameters, hoping to get BLEU (Papineni et al., 2002) scores that are comparable to the Transformer. In order to get a fair comparison, we restrict the search-space for hyperparameters, so that both models use roughly the same amount of parameters. We test both our models on the test set, using Beam Search with a beam size of 5 and alpha of 1.0.

5.1 Training Setup

Both models were trained on one GPU for a total of 3 days. Table 2 shows the hyperparameter settings for both model variants.

Parameter	Transformer	Dec. Only
scheduling	plateu	plateu
optimizer	adam	adam
patience	5	5
LR	0.0003	0.0003
Batch size	16	16
src embedding dim	512	-
trg embedding dim	512	1024
FF size	2048	1024
max sent length	62	62
num encoder layers	6	-
num decoder layers	4	3
num attention heads	4	8
num parameters	57691136	57661440

Table 2: Hyperparameters

6 Results and Discussion

6.1 Hyperparameter Tuning

The Decoder Only System was very sensitive to suboptimal hyperparameter settings which made hyperparameter tuning cumbersome. We can see training progress for different hyperparameter settings in Figure 3. Wider architectures performed better than deeper ones.

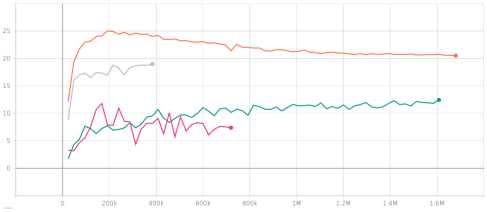

Figure 2: BLEU scores on development set. For different hyperparameter variants of the Decoder Only Model.

6.2 BLEU scores

The Transformer model achieved a BLEU score of 30.19 on the test set. The Decoder Only model achieved a BLEU score of 24.79. The Decoder Only model could not achieve comparable performance to the Transformer.

6.3 Speed

The Transformer model trained faster, processing on average 6477 tokens per second. The Decoder Only Model processed on average 6115 tokens per second. The Transformer showed its biggest speed advantages during validation. An average validation step took the Transformer only 151 seconds

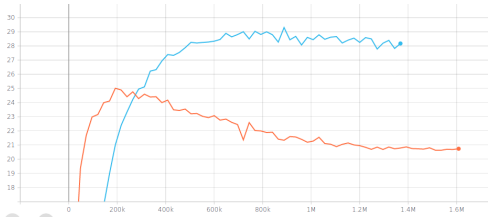

Figure 3: BLEU scores on development set for Transformer (Blue) and Decoder Only (Orange).

while the Decoder only System needed 323 seconds on average. The Transformer is fast at training but slow at inference. During inference the Transformer has to perform as many decoding steps as the output sequence is long. The Decoder Only System suffers from the same problem. However the Transformer has an advantage. The Transformer can use its encoder to process the input sequence in only one step and use the so obtained representation of the input sequence during decoding. In the Decoder Only System, the model has to process the whole input sequence at every decoder step, making it significantly slower than the Transformer.

Model	Tok. per sec	Val. duration
Transformer	6477	151s
Dec. Only	6115	323s

Table 3: Training and validation speed of the two models.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks.

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