Ruprecht-Karls-Universität Heidelberg
Introduction to Neural Networks and Sequence-To-Sequence Learning (SS 23)

Project Report

Replicating the results from 'Attention is All You Need' for English to German translation

Nils Schacknat
September 9, 2023

Abstract

The Transformer architecture, as introduced in the famous 'Attention is All You Need' paper, revolutionized natural language processing by presenting a highly effective approach for sequence-to-sequence tasks. The objective of this project is to gain hands-on experience with this architecture by implementing and applying it to the same translation task used in the original paper. Through cloud-based training, this project replicates the reported results for English to German translation on the 'base' model.

Author: Nils Schacknat
Field of study: Data and Computer Science, MSc.
Matriculation number: 4078555
Lecturer: Michael Staniek
Code Base: https://github.com/nils-schacknat/transformer

1 Introduction

The emergence of the Transformer architecture stands as a major milestone in the field of natural language processing (NLP), as it offers an innovative and highly efficient way to train neural networks for sequence to sequence tasks. Departing from the conventional recurrent approach, the Transformer is trained in parallel on multiple sequence pairs in only one forward pass, resulting in substantial efficiency gains. Central to the functionality of the Transformer is its attention mechanism, which empowers it to manage long-range dependencies, while masked-attention, a specialized variant, enables the ability to train without recurrence.
Originated in 2017 by Vaswani et al. in their famous seminal paper 'Attention is all you need' [1], the Transformer architecture continues to hold its position as state-of-the-art, powering today's exceptional language models such as GPT and LLaMA.

The central objective of this project is to gain familiarity with the Transformer by implementing the architecture as detailed in the original paper. The focus is directed towards creating a functional model and ideally replicating the results reported by the authors. Given their comprehensive evaluation across various datasets and parameter configurations, this project focuses on the English to German translation task using their specified 'base' model. Cloud-based training is employed, in order to match the required computational power. The main goal is to learn about this innovative architecture and to collect my first proper experience with an NLP task.

2 Literature

Naturally, the 'Attention is all you need' paper forms the heart of this project. Whenever this report mentions 'the authors' or 'the paper', it refers to this work.
Additionally, 'The Illustrated Transformer' [2] by Jay Alammar contributed to enhance my overall understanding of the concepts and the PyTorch implementation [3][4] proved to be valuable at times, when the written description was not precise enough, at least from my perspective.

3 Methodology

As this is a 1:1 replication, I won't discuss the details of the architecture here and will instead focus on the practical aspects of the implementation.

The implementation is done in Python. As the goal is to create the architecture from the ground up, only 'plain' PyTorch [5] is used here. This means all Transformer components are implemented using tensor arithmetic and basic structures such as forward layers, embedding layers and layer norm.

The dataset, as well as the tokenizer setup is the same as described in the original paper.
Notable libraries that are used here are:

- Hugging Face's *Datasets* [6] library for the WMT2014 English-German dataset [7].

- Google's *SentencePiece* [8] library for training a byte-pair encoding (BPE) [9] tokenizer.
- *TorchText* [10] and *TorchData* [11] for additional data preprocessing and metric calculations.

The training process is hosted on the Lambda Labs platform [12], utilizing an NVIDIA RTX A6000 GPU with 48GB of memory.

I also made use of ChatGPT 3.5 [13], primarily for code documentation, but also for some coding and writing assistance.

4 Implementation

Here, I discuss the two major challenges encountered during implementation and how I addressed them.

As this is my first proper NLP project, I initially underestimated the importance of the supporting setup, for example concepts like bucketing sequences of similar lengths together to enhance training efficiency. It also took some time to identify the right libraries for different tasks, such as *SentencePiece* for tokenization and *TorchData* + *TorchText* for data transformation and preprocessing. While these libraries were unfamiliar to me at first, after some research and many revisions, I was able to develop a robust setup, suitable for my needs.

When coding the Transformer architecture, I had a basic understanding of its functioning. However, I occasionally encountered specific nuances during the implementation that required a deeper dive into its mechanics. For example, I initially missed the fact that the Transformer generates predictions at every point in the target sequence, a key element contributing to its efficient training. In moments like these and when the original paper didn't provide all the necessary insights, at least from my perspective, I turned to the PyTorch implementation for clarification. Even though this slowed down my progress, it was an important part in enhancing my understanding of the architecture's workings.

Final repository setup

```
├── config.yaml                    # Holds model and training parameters
├── demo_notebooks                 # Notebooks to demonstrate model capabilities
│   ├── translation_demo.ipynb
│   └── visualize_attention.ipynb
├── main.py                        # Starts the training
├── model.py                       # Assembles the final model
├── README.md
├── requirements.txt
├── tokenizer                      # Directory to train and store the tokenizer
│   └── create_tokenizer.py
├── trainer.py                     # Implements the training loop
├── transformer_components.py
├── translation_datapipe.py        # Creates a datapipeline for the dataset
└── util.py
```

Code Base: https://github.com/nils-schacknat/transformer

5 Experimental Setup

The parameter configuration used for the model here, is the same as the configuration of the 'base' model described in the original paper. All training hyperparameters remain consistent as well, with two exceptions, which are highlighted in the table below.

	Train Steps	Batch Size	GPU	BLEU
Base Model	100K	25K	8×P100 (8×16GB)	25.8
My Implementation	50K	20K	RTX A6000 (48GB)	25.9

Table 1: Batch size indicates the approximate number of source and target tokens each. (Padding tokens are not counted.) The GPUs are from NVIDIA.

The batch size was chosen as large as the GPU allowed for. To make up for the smaller memory, the training is done with single precision floats, effectively doubling the GPU's capacity.

To cut down on costs and since the models performance had started to converge (see Figure 1), the number of training steps was limited to 50K, which resulted in a total training time of 11 hours.

Unfortunately, the WMT2014 English-German dataset contains several mismatched sequence pairs. Thus, another modification involves the incorporation of filtering into the preprocessing phase. This might deliver an explanation as to why this implementation was able to match the reported performance, despite using a smaller batch size and fewer training steps.

Sequence pairs were filtered under the following conditions.

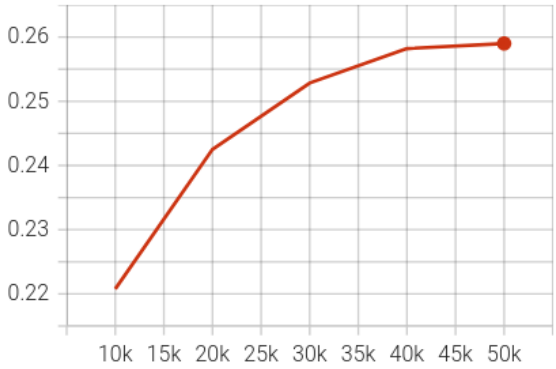
- Source or target sequence to short (sometimes a sentences was matched against an empty line or a single character)
- Source or target sequence to long (just to avoid overly long sentences which drastically increase the amount of padding tokens)
- Unknown characters present (for some reason there were sentences with Chinese characters)
- Relative difference of source and target sequence lengths too large (another indicator for mismatched sentences)

Here is an example where lots of errors can be observed, although in general, the dataset appears to mostly be correct.

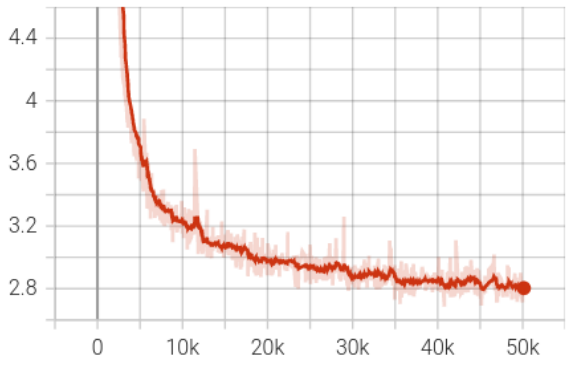
(The filtering of sequence pairs is applied to the training set only.)

6 Results

This implementation scored a BLEU value of 25.9, confidently matching the reported value of 25.8. Next, this section showcases the curves for BLEU score and loss, followed by the presentation of some translation examples.



(a) BLEU score, evaluated every 10K steps.



(b) Smoothed loss.

Figure 1

English	**German (translated)**
Heidelberg is a city at the Neckar in the southwest of Germany.	Heidelberg ist eine Stadt am Neckar im Südwesten Deutschlands.
Fear is the path to the dark side.	Angst ist der Weg zur dunklen Seite.
I love cake.	Ich liebe Kuchen.
Artificial intelligence is the superior field of research	Künstliche Intelligenz ist das überlegene Forschungsfeld
Attention is all you need.	Achtung ist alles, was Sie brauchen.
I really enjoyed pursuing this project.	Ich habe dieses Projekt wirklich verfolgt.
I really liked pursuing this project.	Ich habe dieses Projekt sehr gerne verfolgt.
There's always a bigger fish	Es gibt immer einen größeren Fisch
I am the senate!	Ich bin der Senat!
I find your lack of faith disturbing	Ich finde Ihren Mangel an Glauben beunruhigend.
It's a trap!	Es ist eine Falle!
This sentence serves as an example for showcasing the translation performance of this model.	Dieser Satz dient als Beispiel für die Darstellung der Übersetzungsleistung dieses Modells.

Table 2: Translation examples, as found in `demo_notebooks/translation_demo.ipynb`

The translations may not always be flawless, and it's not too difficult to create sentences where the model struggles, but overall, the performance is quite respectable.

7 Conclusion

In summary, this project has been successful, both in terms of practical results and learning outcomes. Firstly, I managed to closely replicate the results reported in the original paper, which is a nice achievement in itself, but also serves as a good validation for my implementation. Second, this project allowed me to gain hands-on experience with cloud computing, broaden my knowledge of natural language processing, and, most importantly, significantly deepen my understanding of the Transformer architecture − fulfilling the initial motivation behind this assignment.

References

[1] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[2] Jay Alammar. *The Illustrated Transformer*. URL: https://jalammar.github.io/illustrated-transformer/.

[3] PyTorch. *Language Translation with nn.Transformer and torchtext*. URL: https://pytorch.org/tutorials/beginner/translation_transformer.html.

[4] PyTorch. *PyTorch source of nn.Transformer*. URL: https://pytorch.org/docs/stable/_modules/torch/nn/modules/transformer.html.

[5] *'PyTorch' Library*. URL: https://pytorch.org/.

[6] Hugging Face. *'Datasets' Library*. URL: https://huggingface.co/docs/datasets/index.

[7] Ondrej Bojar et al. "Findings of the 2014 Workshop on Statistical Machine Translation". In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Baltimore, Maryland, USA: Association for Computational Linguistics, June 2014, pp. 12–58. URL: http://www.aclweb.org/anthology/W/W14/W14-3302.

[8] Google. *'SentencePiece' Library*. URL: https://github.com/google/sentencepiece/.

[9] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural machine translation of rare words with subword units". In: *arXiv preprint arXiv:1508.07909* (2015).

[10] *'TorchText' Library*. URL: https://pytorch.org/text/stable/index.html.

[11] *'TorchData' Library*. URL: https://pytorch.org/data/beta/index.html.

[12] LambdaLabs. URL: https://lambdalabs.com/.

[13] OpenAI. *ChatGPT 3.5 (Aug 3 version)*. https://www.openai.com.