A Journal on Training an LLM from Scratch

Leonard Cseres

October 06, 2025

Contents

Introduction

I starting working on this project with the interest to reproduce what code completion large language models are capable of doing. My goal is to optimize a tiny completions model, and discover where are the limits.

This journal serves as documentation for the design decisions, implementation choices, parameter changes, etc.

Disclamer: This is an education project and there will be mistakes. Contributions are welcome!

Week 22.09.25

Since we are building a new model from scratch, I wanted to also retrain a new tokenizer. In addition to learning how all this works, I wanted to make changes in the regex parsing, which groups tokens together.

Building the Tokenizer

I decided to reimplement BPE (byte-pair-encoding) tokenization. The main purpose was to learn the BPE algorithm and control it.

I know I needed a high-performance language for this take, so I started out implementing a version in C. However, I quickly realized that the algorithm depends on quite a lot of datastructures (vectors, hashmaps, heaps) in addition to the regex parsing. Therefore, I switched to C-style C++.

For the training data, I trained the tokenizer exclusively on Python code using the `py150` datasetrained the tokenizer exclusively on Python code using the `py150` dataset.

Since python code is *usually* written using ASCII characters, I removed UTF-8 characters before tokenizing, effectively excluding them from the vocabulary. This removes tokens that are very rarely used and allows the model to focus on the essential.

Creating the Regex

I took inspiration from the cl100k_base regex [1]:

```
'(?i:[sdmt]|ll|ve|re)|[^\r\n\p{L}\p{N}]?+\p{L}++|\p{N}{1,3}+| ?[^\s\p{L}\p{N}]+
+[\r\n]*+|\s++$|\s*[\r\n]|\s+(?!\S)|\s
```

and ended up on the following regex:

```
 ?[A-Za-z_(][A-Za-z_.]*|%(?:\.\d+)?[sdifFeEgGxXoc%]|[0-9]{1,3}| ?[^ %_A-Za-
z0-9]+(?: ")?[\r\n]*|%|\s+$|\s+(?=\s)|\s
```

- I removed UFT-8 handling and compound expression groupping (ex: `'ve`)
- ` ?[A-Za-z_(][A-Za-z_.]*` groups together charaters with `_` , `(` , and `.` symbols
- `%(?:\.\d+)?[sdifFeEgGxXoc%]` groups together printf formats like `%s` and the rest says very similar

Week 29.09.25

Building the Training Loop

I used the `torch` along with `lightning` library [2] to implement a training loop.

For the model, instead of reimplementing from scratch I used the Qwen3 (without MoE) model from the `transformers` library [3].

I choose AdamW for the optimizer with a warmup and cosine schedule. Below is a non exaustive list of the parameters chosen:

```yaml
data:
  split_ratio: 0.7
  vocab_size: 20260 # 256 byte tokens + 20000 BPE merges + 4 special tokens
(BOS, EOS, PAD, UNK)

model:
  # Qwen3 architecture parameters
  hidden_size: 512
  num_hidden_layers: 4
  num_attention_heads: 16
  num_key_value_heads: 8
  intermediate_size: 1024
  max_position_embeddings: 2048
  rope_theta: 10000.0
  attention_dropout: 0.1
  rms_norm_eps: 0.000001

training:
  batch_size: 32
  epochs: 50
  lr: 0.0001
  weight_decay: 0.01
  grad_clip: 1.0
  gradient_accumulation_steps: 4
```

Position IDs & Attention Mask

TODO

Week 06.10.25

BOS Special Token

I added this "beginning of sequence" speical token to allow it to act as a "attention skin" [4].

The `BOS` token is added at the start of every input sequence to the model. Is is not like the `EOS` (end of sequence) token, where it is added between documents to delimit them.

```
input = [BOS, 234, 6236, 346, 4357, 347, ...]  # where BOS is the token id for
BOS
```

Training Fixes

I updated the scheduler to step on every step instead of epoch. I also scaled the training loss over the accumulated batches instead of using the loss of the step.

Run #1

In progress

Bibliography

[1] OpenAI, “cl100k_base tokenizer regular expression in tiktoken_ext/openai_public.py, line 89.” 2025.

[2] Lightning, “Lightning GitHub.” 2025.

[3] Huggingface, “Transformers GitHub.” 2025.

[4] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis, “Efficient Streaming Language Models with Attention Sinks.” [Online]. Available: https://arxiv.org/abs/2309.17453