Building a Miniature Transformer Language Model from Scratch

Kartik Virmani

November 11, 2025

Abstract

This report describes the end-to-end implementation of a small decoder-only Transformer trained as a language model. The model was trained on a corpus composed of three classical English texts: *The Hound of the Baskervilles*, *Shakespeare's Works*, and *War and Peace*. We implement every component of the network from scratch in PyTorch, including Byte-Pair Encoding (BPE) tokenization, positional embeddings, causal multi-head self-attention, feedforward MLPs, and layer normalization. The model learns to predict the next token given the previous context and can generate coherent sentence completions auto-regressively.

1 Dataset and Tokenization

The dataset consists of concatenated text from:

- `hounds_of_baskervilles.txt`
- `shakespeare.txt`
- `war_and_peace.txt`

We used the HuggingFace `tokenizers` library to train a Byte-Pair Encoding (BPE) tokenizer with a vocabulary size of 2048. The tokenizer includes several special tokens:

`[PAD]`, `[UNK]`, `[CLS]`, `[SEP]`, `[MASK]`

The preprocessing pipeline included:

- **BertNormalizer:** lowercasing, accent stripping, and whitespace cleaning.
- **BertPreTokenizer:** splitting contractions and punctuation.
- **TemplateProcessing:** to append `[CLS]` and `[SEP]` for each sentence.

The tokenizer was trained as:

```
bpe_model = models.BPE(unk_token="[UNK]")
tokenizer = Tokenizer(bpe_model)
trainer = trainers.BpeTrainer(vocab_size=2048, min_frequency=2,
                              special_tokens=["[PAD]","[UNK]",
                              "[CLS]","[SEP]","[MASK]"])
tokenizer.train(files=[...], trainer=trainer)
```

2 Model Architecture

We implemented a decoder-only Transformer consisting of:

- Embedding dimension $d = 256$
- Context length $T = 128$
- Two attention heads ($h = 2$), each with key/query/value dimension $p = 32$
- Two to four Transformer blocks with layer normalization and residual connections
- A position-wise MLP with hidden size $2d$

2.1 Positional Encoding

Sine-cosine positional encoding was implemented as:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

This encoding was added to the token embeddings to inject order information.

2.2 Self-Attention Mechanism

Each head computes:

$$q_t = x_t W_q, \quad k_t = x_t W_k, \quad v_t = x_t W_v$$

and attention scores

$$\text{Attn}(q_t, k, v) = \text{softmax}\left(\frac{q_t k^\top}{\sqrt{p}} + M_{\text{causal}}\right) v$$

where M_{causal} is a triangular mask enforcing causality. Key-padding masks were also applied to ignore padded tokens.

Outputs from multiple heads were concatenated and projected:

$$h' = [\text{head}_1; \text{head}_2] W_o$$

2.3 Residual and MLP Layers

Each block used the pre-norm form:

$$h = h + \text{Attention}(\text{LayerNorm}(h)), \quad h = h + \text{MLP}(\text{LayerNorm}(h))$$

where the MLP consisted of two linear layers with ReLU activations:

$$\text{MLP}(x) = \text{ReLU}(xW_1)W_2$$

3 Training Procedure

We trained the model on next-token prediction:

$$\mathcal{L} = -\sum_t \log p(x_t | x_{<t})$$

using `CrossEntropyLoss` with padding tokens ignored. Each mini-batch contained variable-length sentences padded to the longest sequence within the batch.

- Optimizer: AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, lr=3×10^{-4})

- Weight decay: 0.01
- Gradient clipping: 1.0
- Batch size: 64
- Evaluation: validation loss every 1000 updates

3.1 Loss Curves

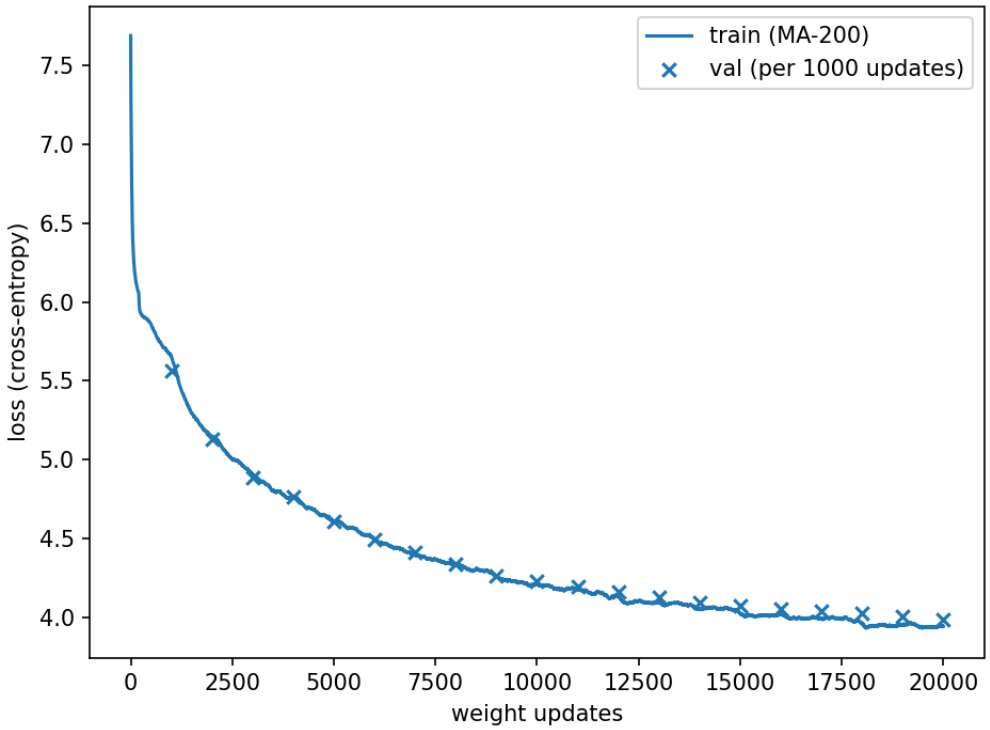

Figure 1: Training loss (moving average) and validation loss. The loss decreases steadily, confirming convergence.

4 Auto-Regressive Sampling

We implemented auto-regressive sampling using:

$$\hat{x}_{t+1} \sim p_\theta(\cdot \mid x_1, x_2, \ldots, x_t)$$

At each step, we sample from the model's predicted logits using temperature scaling and top-k filtering:

$$p_i = \frac{\exp(z_i/\tau)}{\sum_j \exp(z_j/\tau)}$$

with $\tau = 0.9$ and $k = 40$.

4.1 Examples of Generated Text

```
PROMPT: The Eiffel Tower is in Paris
GEN   : the e if fe l to wer is in paris
```

```
PROMPT: Sherlock Holmes looked at the
GEN   : s her lock hol mes looked at the at least , he looked down at her

PROMPT: To be or not to
GEN   : to be or not to for my love the duke
```

4.2 Interpretation

- The fragmented words ("e if fe l") arise from fine-grained BPE merges due to a small vocabulary size.
- The model captures grammatical structure and continuity, demonstrating that causal self-attention and positional encoding were implemented correctly.
- Increasing vocabulary size or training iterations would improve lexical coherence.

5 Conclusion

This project implemented a minimal large language model entirely from first principles. Despite a small dataset and limited vocabulary, the network successfully learned context-dependent token prediction and generated syntactically valid text continuations. Future extensions include:

- Expanding vocabulary to 8K–16K tokens
- Adding more layers and training steps
- Introducing cosine learning-rate scheduling and mixed precision
- Scaling to larger corpora for coherent paragraph-level generation

References

[1] Sennrich, R., Haddow, B., and Birch, A. (2016). *Neural Machine Translation of Rare Words with Subword Units*. ACL.

[2] Vaswani et al. (2017). *Attention is All You Need*. NIPS.

[3] HuggingFace Tokenizers: `https://huggingface.co/docs/tokenizers/quicktour`