From Transformer to Large Language Models

Applied Deep Learning — Chapter 11

Yin Yang

Foundation Books Project

Roadmap

Decoder-Only LLMs

Tokenization

Language Modeling Objective

Inside a Decoder-Only Transformer

Decoding

When Inference Is Not Enough

A Practical LLM Recipe

Summary

Decoder-Only LLMs

The Running Example

Study assistant: a learner asks a question, the system answers.

- Prompt tokens $(x_1, \ldots, x_T)$.
- Model scores possible x_{T+1}.
- Decoding rule chooses one; append; repeat.

An answer can be fluent and wrong because the prompt template, retrieval, decoding rule, or context budget was wrong — not because the weights are bad.

Autoregressive Language Model

An autoregressive language model factors:

$$p_\theta(x_1, \ldots, x_T) = \prod_{t=1}^{T} p_\theta(x_t \mid x_{<t}).$$

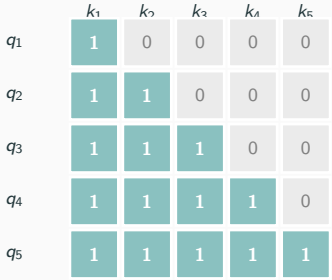
Each token uses only earlier tokens.

Inside a Transformer, this rule is enforced by a causal mask.

$$M_{ij} = \begin{cases} 0 & j \le i, \\ -\infty & j > i. \end{cases}$$

Adding M to the attention scores before softmax makes future positions invisible.

Causal Mask Visible Pattern (T=5)



Token 4 reads itself and earlier, but not token 5. Same rule whether the input is English, code, or mixed scripts.

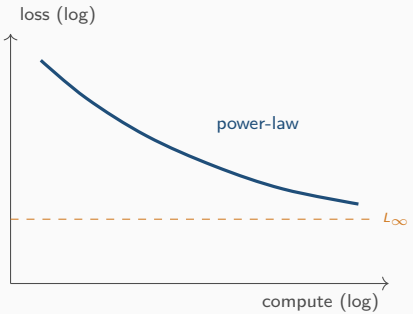
Why Decoder-Only Took Over

- GPT-1: decoder-only + supervised fine-tuning.
- GPT-2: scale, multitask behavior.
- GPT-3: few-shot from prompts alone.

Empirical scaling:

$$L(N, D) \approx L_\infty + aN^{-\alpha} + bD^{-\beta}.$$

Both N (params) and D (tokens) matter; making only one larger wastes compute (Chinchilla).



Larger is not automatically better for a fixed validation set.

Context Window

The context window is the maximum number of *tokens* a model accepts in one forward pass.

- It's a token budget, not a word budget.
- Prompt + retrieved notes + chat template + answer all share it.
- Longer context $\Rightarrow$ more attention work, larger KV cache.

Longer context cannot fix poor retrieval or contradictory examples.

Tokenization

Subword Tokenization

Modern LLM tokenizers use subword units:

- **BPE** (byte-pair encoding): learn frequent merges.
- **SentencePiece:** train directly on raw text without language-specific pre-tokenization.

Rare words split into pieces rather than becoming one "unknown" token.

Example: `validation accuracy` might split as `validation`, `accur`, `acy` — 3 tokens, not 2 words.

Tokenizer = Checkpoint Contract

Token ID 142 only has the meaning the checkpoint learned.

- Swapping the tokenizer changes the meaning of every input ID.
- Special tokens (BOS, EOS, pad, role separators) carry structural meaning.
- Chat templates wrap user/assistant turns into the exact tokenizable text used during instruction tuning.

Use `apply_chat_template` for inference; truncation and special tokens are visible only after tokenization.

Language Modeling Objective

Logits, Softmax, Cross-Entropy

Output at position t: logits $z_t \in \mathbb{R}^{|V|}$.

Softmax over the vocabulary:

$$p_\theta(x_{t+1} = v \mid x_{\le t}) = \frac{\exp(z_{t,v})}{\sum_u \exp(z_{t,u})}.$$

Next-token cross-entropy if true target is y:

$$\ell(z, y) = -\log\left(\frac{\exp(z_y)}{\sum_u \exp(z_u)}\right).$$

Average across many positions during pretraining.

Numerical Example

Logits over $\{\texttt{Start}, \texttt{Use}, \texttt{Increase}\}$: $z = (2.0,\ 1.0,\ 0.0)$.

$\exp(z) = (7.389, 2.718, 1.000)$, sum 11.107.

$$p \approx (0.665,\ 0.245,\ 0.090).$$

- Target `Start`: $\ell = -\log 0.665 \approx 0.408$.
- Target `Use`: $\ell = -\log 0.245 \approx 1.407$.

Same logits, different loss. Loss rewards matching observed data, not abstract truth.

Perplexity

Average cross-entropy $\bar{\ell}$ exponentiated:

$$\text{PPL} = \exp(\bar{\ell}).$$

Often interpreted as "effective number of choices per token."

Example: $\bar{\ell} = 1.2 \Rightarrow \text{PPL} \approx 3.32$.

Caveat: perplexity depends on tokenizer and dataset. Two models with different tokenizers cannot be compared directly on the same raw text.

Low loss $\neq$ truthful, useful, or well-formatted. Task evaluation is still required.

Inside a Decoder-Only Transformer

Residual Stream

Layer ℓ holds token representations:

$$H^{(\ell)} = (h_1^{(\ell)}, \ldots, h_T^{(\ell)}), \quad h_t^{(\ell)} \in \mathbb{R}^d.$$

Each block *adds* updates:

$$H^{(\ell+1)} = H^{(\ell)} + \text{block updates}.$$

Useful axes to track:

- layer ℓ, batch item, head h, position t, residual width d, head dim d_h.

Shape checks catch most attention bugs before training.

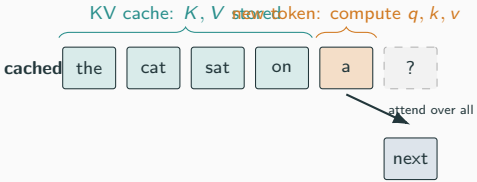
Rotary Position Embeddings (RoPE)

Many modern decoders rotate Q and K coordinate pairs by a position-dependent angle:

$$\begin{bmatrix} q'_{2r} \\ q'_{2r+1} \end{bmatrix} = \begin{bmatrix} \cos(m\omega_r) & -\sin(m\omega_r) \\ \sin(m\omega_r) & \cos(m\omega_r) \end{bmatrix} \begin{bmatrix} q_{2r} \\ q_{2r+1} \end{bmatrix}.$$

- m: token position; ω_r: per-coordinate frequency.
- Relative position appears inside dot-product scores.
- Position is part of the checkpoint — swapping context length or tokenizer is not free.

KV Cache



Toy decoder: 12 layers, 8 heads, head dim 64, context 1024, batch 1.
$12 \times 2 \times 8 \times 1024 \times 64 = 12{,}582{,}912$ numbers ($\approx$ 25 MB at fp16).

- Cache trades memory for speed.
- Doubling context $\Rightarrow$ doubling cache.
- Quantizing weights does *not* shrink the activation cache.

Decoding

From Distributions to Text

Decoding turns logits into text, one step at a time.

Greedy:

$$\hat{x}_{t+1} = \arg\max_{v} p_\theta(v \mid x_{\leq t}).$$

Deterministic; brittle for open-ended answers.

Beam search: keep B partial continuations. Helpful for constrained outputs (translation), often produces generic open-ended text.

Temperature, Top-k, Top-p

Temperature rescales logits before softmax:

$$p_T(v) = \frac{\exp(z_v/T)}{\sum_u \exp(z_u/T)}.$$

$T < 1$: sharper. $T > 1$: flatter.

Top-k: keep the k highest-probability tokens, renormalize.

Top-p / nucleus: smallest set with cumulative probability $\geq p$.

Decoding settings are part of the system, not decoration.

Decoding Numerical Walkthrough

Probabilities for the first answer token:

Token	`Start`	`Use`	`Increase`	`Maybe`	`Banana`
p	0.42	0.30	0.16	0.10	0.02

- Greedy: `Start`.
- Top-2 sample: `Start` (0.583), `Use` (0.417).
- Top-p at 0.80: {`Start`, `Use`, `Increase`} since $0.72 < 0.80 \leq 0.88$.
- `Banana` eliminated by both top-k and top-p.

Stops and Repetition

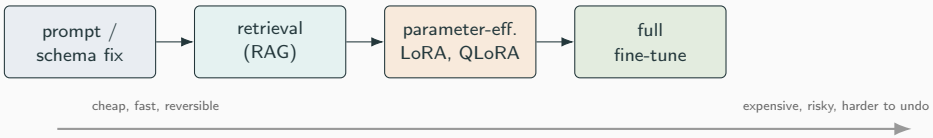
- **Stop sequences:** text patterns that end generation (e.g. a new `Question:` marker, EOS token).
- **Repetition penalty:** reduce score of already-seen tokens or n-grams.
- **Max-new-tokens:** hard upper bound on answer length.

Greedy decoding alone does not guarantee a structural rule like "exactly three numbered recommendations." Without a stop sequence, even greedy can use the entire token budget.

Match decoding to the job: factual extraction $\rightarrow$ low T; brainstorming $\rightarrow$ higher T.

When Inference Is Not Enough

Adaptation Tools, in Order



Fine-tuning changes weights and should answer a specific failure:

1. **Prompt or output-schema fix:** format / verbosity issues.
2. **Retrieval (RAG):** answer needs facts not in pretraining.
3. **Parameter-efficient (LoRA, QLoRA):** stable style or task behavior across many examples.
4. **Full fine-tuning:** large data, compute, evaluation discipline; risk of catastrophic forgetting.

If the failure is decoding or formatting, weight updates make the mistake more expensive.

Catastrophic Forgetting

- Adapting to a new dataset can damage previously good behavior.
- Small study-assistant data + full fine-tune: model memorizes training answers, gets worse on new questions.
- Parameter-efficient methods constrain how far weights can drift.

Before changing weights:

- record the inference baseline (model, template, decoding, validation rubric);
- ask which specific failure the weight update fixes.

A Practical LLM Recipe

Library Roles

Object	Role
`AutoTokenizer`	tokenizer, special tokens, chat template.
`AutoModelForCausalLM`	decoder-only LM weights, device map, precision.
`model.generate`	autoregressive decoding loop with sampling controls.
PyTorch + `accelerate`	devices, kernels, precision, memory measurement.

Record checkpoint, tokenizer, prompt template, decoding settings, hardware, peak memory, package versions.

Reference Decoding Run

`Qwen2.5-0.5B-Instruct`, max-model-len 1024, 96 new tokens, 3 prompts, 2 runs each.

Setting	Runs	1st-tok ms	Tok/s	Unique/prompt
Greedy, $T = 0$	6	12.4	142.5	1.0
$T = 0.7$, top-p=0.9	6	28.6	136.0	2.0

Greedy duplicates were identical; sampled runs produced two different answers per prompt. Quality and cost belong on the same page.

Model-Size Ablation

Same prompts, same decoding, different model:

- `Qwen2.5-0.5B-Instruct`: 0.013 s to first token, 135 tok/s.
- `Qwen2.5-1.5B-Instruct`: 0.027 s to first token, 48 tok/s.

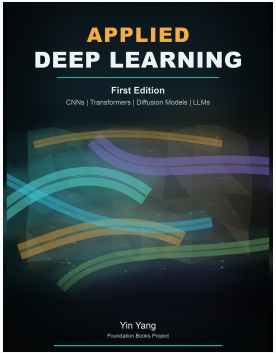
The larger model has to earn the extra latency on the same rubric — being bigger is not the same as being better here.

Summary

Summary

- **Decoder-only LLMs** predict the next token from earlier tokens with a causal mask.
- **Tokenization** is part of the model contract; subwords, special tokens, and chat templates change input meaning.
- **Objective:** softmax cross-entropy averaged over next-token positions; perplexity is exponentiated cross-entropy.
- **Internals:** residual stream, RoPE-style positional info, KV cache trade memory for speed.
- **Decoding:** greedy, beam, temperature, top-k, top-p, stops — part of the system.
- **Adaptation order:** prompt $\rightarrow$ retrieval $\rightarrow$ parameter-efficient $\rightarrow$ full fine-tune.

Next: retrieval-augmented generation — chunking, dense and lexical retrieval, hybrid + reranking, evaluating answers and citations.



Applied Deep Learning

Chapter overview slides for the book by Yin Yang.

Book page	`foundation-books.github.io/applied-deep-learning`
Code	`github.com/foundation-books/applied-deep-learning-code`
Print	`amazon.com/dp/B0GZF7QRSH`
Kindle	`amazon.com/dp/B0GZF9221X`
License	CC BY-NC-ND 4.0

These free overview slides may be shared non-commercially with attribution and without modifications. Full explanations, exercises, and companion-code context are in the book and linked repository.