SAMIR GENAI ACADEMY

PROFESSIONAL STUDY GUIDE SERIES

Phase 01 · GenAI Foundations

02

How LLMs Work

Transformers, attention, tokenization & emergent intelligence

Module 02 of 25 · **1 hour**

Instructor: **Samir Kumar** · Edition 2026

STUDY GUIDE

01 Overview

> Transformers, attention, tokenization & emergent intelligence. This guide gives you the core theory, the tools that matter, a hands-on build, working code, and the pitfalls to avoid — everything you need to learn the topic to a professional standard and apply it in real projects.

1 hour	4 + 4	3	02/25
LESSON LENGTH	CONCEPTS + BUILD STEPS	FRAMEWORKS & TOOLS	PHASE 01

02 Learning Objectives

- Explain tokenization and why token counts drive cost and context limits.
- Describe the transformer block: embeddings, self-attention, and feed-forward layers.
- Understand how next-token prediction produces emergent capabilities.
- Manage context windows deliberately for long inputs.

03 Core Concepts

Tokenization

Text is split into tokens (sub-word units) before a model sees it. A token is roughly 4 characters of English. Token counts determine cost, latency, and whether input fits the context window. Code, non-English text, and rare words tokenize less efficiently — always measure rather than guess.

Embeddings & attention

Each token becomes a high-dimensional vector (embedding). Self-attention lets every token weigh the relevance of every other token, capturing long-range dependencies — this is the core innovation of the transformer. Multi-head attention runs several attention patterns in parallel, each learning different relationships.

Next-token prediction

An LLM is trained to predict the next token given all previous tokens. Despite this simple objective, scaling parameters and data yields emergent abilities — reasoning, translation, code generation — that were never explicitly programmed. Generation is autoregressive: each predicted token feeds back as input.

Context management

The context window is the model's working memory. Long documents must be chunked, summarised, or retrieved (RAG, covered later). Position within context matters — critical instructions belong near the top or bottom where models attend most reliably.

04 Visual Diagram

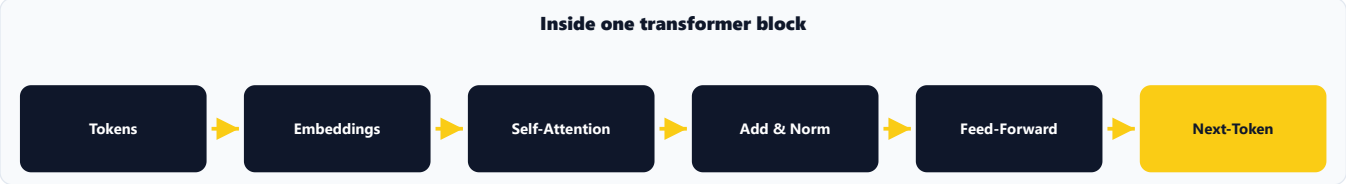

Figure 2 — Data flow through a transformer block. Self-attention lets every token weigh every other token before the feed-forward layer refines the representation.

05 Formula Spotlight

Scaled dot-product attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(Q \cdot K^T / \sqrt{d_k} \right) \cdot V$$

Q, K, V	query, key, value matrices (projected token embeddings)
$Q \cdot K^T$	similarity score between every pair of tokens
$\sqrt{d_k}$	scaling by key dimension keeps gradients stable
`softmax`	turns scores into attention weights that sum to 1

Each token builds its new representation as a weighted blend of all value vectors, where the weights come from how relevant every other token is. This single operation is what gives transformers their long-range context.

06 Key Frameworks & Tools

tiktoken / tokenizers | Hugging Face Transformers | Attention-visualisation tools (BertViz)

07 Hands-On Build

🛠 Transformer Architecture Walkthrough + Context-Management Strategy

1. Tokenize the same sentence in English, code, and another language; compare token counts.
2. Trace one transformer block on paper: embedding -> attention -> add&norm -> FFN.
3. Measure how prompt length affects latency and cost across three input sizes.
4. Design a chunking + summary strategy for a 50-page document.

08 Code Walkthrough

Counting tokens before sending a request — PYTHON

```
import tiktoken

enc = tiktoken.get_encoding("cl100k_base")
text = "Transformers use self-attention to model relationships between tokens."
tokens = enc.encode(text)
print(f"{len(tokens)} tokens")
print(tokens[:10])
```

09 🧪 Lab Assistant

Lab 2 · Tokenisation & context measurement

Prerequisites:pip install tiktoken

1. **Encode an English sentence, a code line, and a non-English sentence**
   > Expected: Token counts differ markedly for the same character length
2. **Print the first 10 token ids**
   > Expected: A list of integers — the model's actual input
3. **Estimate cost = tokens × price-per-token**
   > Expected: A per-request cost you can budget with

Troubleshooting

- Unknown encoding → use 'cl100k_base'.
- Counts look off → remember 1 token ≈ 4 English characters, not 1 word.

10 Pitfalls & Key Takeaways

⚠ COMMON PITFALLS

- Assuming 1 word = 1 token (it varies by language and content).
- Stuffing the whole document into context when retrieval would be cheaper and better.
- Burying key instructions in the middle of a very long prompt.

✓ KEY TAKEAWAYS

- Self-attention is the transformer's core mechanism for context.
- Token counts govern cost, latency, and context fit — always measure.
- Capabilities emerge from scale on a simple next-token objective.

📺 **Companion video:** Watch the module 02 lesson in your Student Dashboard, then use this guide to revise, copy the code, and complete the build. Download both for offline study.

11 Further Resources

'Attention Is All You Need' (Vaswani et al.) • The Illustrated Transformer — Jay Alammar • Hugging Face NLP Course