Your changes have been published.

Medium Search Write

Tokenization + Algebra: How Language Models Learn to Count, Classify, and Compute

Francisco Revelles 4 min read · Sep 26, 2025

By Francisco Revelles

Author's Note

This article represents a refined version reviewed by both **GPT-5** and **Gemini**, two AI evaluators used to test conceptual clarity and mathematical framing. It distinguishes analogy from mechanism, defines terminology precisely, and incorporates supporting references to contemporary theoretical work.

A Note on This Framework

The ideas presented here offer a **conceptual lens** for interpreting large language models through algebra.
This is an *analogy* intended to build intuition — not a literal description of algorithms or a proven mechanistic model.

Introduction

Your changes have been published.

Your changes have been published.

TOKENIZATION + ALGEBRA

HOW LANGUAGE MODELS LEARN TO COUNT, CLASSIFY, AND COMPUTE

Your changes have been published.

The graphic highlights the central parallels between algebra and language models: polynomials with token sequences, and modular addition with attention.

Language Models (LLMs) aren't just statistical parrots — they can also be understood as **algebraic machines.** Each token they process behaves like an element within a structured system of relationships, and the operations that connect those tokens resemble the arithmetic logic behind **cryptography, signal processing, and number theory**.

Throughout this article, algebra serves as an interpretive lens connecting symbolic language units with the mathematical reasoning patterns that emerge in neural architectures.

In short, **tokenization is more than preprocessing — it is a gateway to algebra.** From token IDs to embeddings, from attention matrices to emergent reasoning, the architecture of LLMs can be viewed through the lens of **modular arithmetic**.

1. Tokenization as a Gateway to Algebra

Tokenization maps language into a finite alphabet of symbols — much like **elements of a finite field**, such as integers modulo a prime p ($\mathbb{Z}_p$).

- Tokens are **atomic**, each with a unique integer ID.
- Token embeddings are **vectors in continuous space**, yet their transformations behave in modular or categorical ways.

- Sequences of tokens resemble **polynomials over finite fields**, providing a

Your changes have been published.

For example, the mapping from tokens to IDs (see Figure 1) mirrors how elements in $\mathbb{Z}_p$ index positions in modular arithmetic.
Seen this way, the building blocks of language are not arbitrary — they are **algebraic structures in disguise.**2. Modular Arithmetic Inside LLMs

2. Modular Arithmetic Inside LLMs

Empirically, we cannot yet prove these parallels; however, as a **conceptual model**, modular arithmetic illuminates how information might circulate through a network.

As discussed in *How Modular Arithmetic Powers Everything from AI to Cryptography*, LLMs may perform operations that resemble algebraic computation:

- Hidden layers perform transformations that **may** correspond to modular addition or multiplication.
- Embedding spaces and attention matrices **may exhibit** periodic or cyclic behavior analogous to finite groups.
- The “wrap-around” behavior in transformers resembles arithmetic rollover, where values repeat modulo n.

These are not proven mechanisms — but they provide a powerful way to think about how symbolic structures persist through continuous spaces.3.

The Algebraic View of Model Layers

Your changes have been published.

3. The Algebraic View of Model Layers

We can visualize the model's structure as a set of **algebraic lifts** — each stage translating discrete symbols into richer mathematical representations.

- **Helix metaphor:** imagine each token's representation spiraling upward through layers like a helix, marking conceptual phase shifts in meaning.
- **Lexical layer:** discrete tokens and subwords ($\mathbb{Z}_p$).
- **Embedding layer:** continuous vector basis ($\mathbb{R}^n$).
- **Attention layers:** rotational transformations analogous to modular arithmetic.
- **Output projection:** grounding back into discrete language.

Your changes have been published.

Figure 1 — The Tokenization → Algebra Pipeline
Language tokens are lifted from $\mathbb{Z}_p$ (discrete) into $\mathbb{R}^n$ (continuous), transformed by modular-like rotations, and projected back into symbols.

4. Practical Implications for LLM Design

This algebraic perspective has design consequences:

- **Tokenizer optimization:** Subword tokenizers can act like "prime factor

Your changes have been published.

- **Algebra-aware architectures:** Embedding spaces constrained to algebraically inspired lattices could improve modular reasoning.
- **Error diagnosis:** Misalignments in arithmetic analogy often correlate with hallucination, suggesting new tools for interpretability.

By treating language as algebra, we can design models that reason more robustly and fail more predictably.

Your changes have been published.

Your changes have been published.

This diagram illustrates how the internal mechanics of large language models (LLMs) mirror algebraic structures. Each stage in an LLM — from tokenization to attention — corresponds to a classical algebraic operation: discrete math (finite fields, $\mathbb{Z}_p$) maps to tokenization, polynomials to embeddings ($\mathbb{R}^n$), modular addition to attention transformations, and algebraic closure to output generation.

5. Toward a General Theory of Token Algebra

Across both theory and practice, these ideas point toward a unifying principle:

> **LLMs may be fruitfully interpreted as algebraic processors** —
> discrete tokens lifted into continuous embeddings, transformed through modular rotations, and projected back into language.

This framing bridges:

- **Symbolic AI** — discrete logic and reasoning.
- **Deep Learning** — continuous optimization.
- **Pure Mathematics** — field theory and modular arithmetic.

By making this algebraic nature explicit,
we open a path toward architectures that reason **as elegantly as they generate.**

Your changes have been published.

Figure 3. The Algebraic Stack — Conceptual schematic of the representational hierarchy in a language model. Lexical tokens and subwords originate as discrete elements in the finite field $\mathbb{Z}_p$ and are lifted into continuous vector space $\mathbb{R}^n$ through embedding layers. Attention layers perform rotations and modular transforms within $\mathbb{R}^n$ to capture compositional structure, and the output probability layer projects the transformed representation back into linguistic space.

Cl Your changes have been published.

By viewing language as algebra, we shift from seeing LLMs as opaque statistical systems to seeing them as **computational engines with internal, learnable logic.** Recognizing these parallels between algebra and cognition allows us to design models that **count, classify, and compute** with greater transparency.

Future essays will extend this framework toward **empirical visualization of token-space periodicity**.

Figures

Figure 1 — The Tokenization → Algebra Pipeline
Figure 2 — Attention as Modular Arithmetic
Figure 3 — The Algebraic Stack: a vertical flow from Discrete Tokens ($\mathbb{Z}_p$) → Embeddings ($\mathbb{R}^n$) → Attention ("rotations") → Output Probabilities.

References

- Cohen & Welling (2016). *Group Equivariant Convolutional Networks.*
- Tancik et al. (2020). *Fourier Features Let Networks Learn High-Frequency Functions.*
- Bronstein et al. (2021). *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics.*

- Token Topology: A Geometric View of Language Representations (2024),

Your changes have been published.

Related Articles — The Trilogy

1️⃣ *What Finite Fields Can Teach Us About Building Smarter Language Models*
2️⃣ *How Modular Arithmetic Powers Everything from AI to Cryptography*
3️⃣ **Tokenization + Algebra: How Language Models Learn to Count, Classify, and Compute**

Tags

#AI #LanguageModels #Algebra #MachineLearning #Interpretability #Tokenization #ModularArithmetic #Cognition

Artificial Intelligence | Language Models | Mathematics | Machine Learning Theory | Modular Arithmetic

Written by Francisco Revelles

2 followers · 38 following

Designing intelligent systems with clarity and purpose. AI, software development, and the art of elegant code.

Edit profile

No responses yet

Your changes have been published.

Francisco Revelles

What are your thoughts?

More from Francisco Revelles

Francisco Revelles

LLM Agents and Service Dogs: How Artificial Intelligence Learns to...

By Francisco Revelles

1d ago

Francisco Revelles

Cognitive Shuffling: How a Sleep Trick Reveals the Logic of AI and...

Have you ever gone to bed with your mind racing, unable to turn off the noise? Your...

Oct 2

In Artificial Intelligence in Plain... by Francisco Re...

How Modular Arithmetic Powers Everything from AI to...

By Francisco Revelles

Sep 26

Francisco Revelles

What Finite Fields of Size p Can Teach Us About Building Smarter...

By Francisco Revelles

Jun 20 13

See all from Francisco Revelles

Recommended from Medium

In Towards AI by M

Connie Zhou

Apple's Approach to Large La

Your changes have been published.

An analysis of Apple's AI capabilities and limitations.

5d ago

Mastering Natural Language

Natural Language Processing (NLP) has seen remarkable progress over the past decade,...

Jun 30 1

Nishikant

How to Optimize RAG Context Windows for Smarter Retrieval

Learn how RAG context windows shape retrieval pipelines with multi-stage retrieval,...

5d ago

Ayhan Bozkurt

Navigating the AI Model Landscape: From LLMs to MoEs—...

Artificial intelligence is everywhere, but what lies beneath the surface? Understanding the...

Oct 2 2

Donder

AI-Powered App Localization: From Weeks to Minutes with Localit.io

How we built an AI translation system that localizes apps 10x faster than traditional...

Satyam Mishra

Causal Fields: Modeling Transformer Attention as Quantu...

How Quantum Field Theory Can Explain Context Propagation, Causality, and...

Jun

Your changes have been published.

See more recommendations