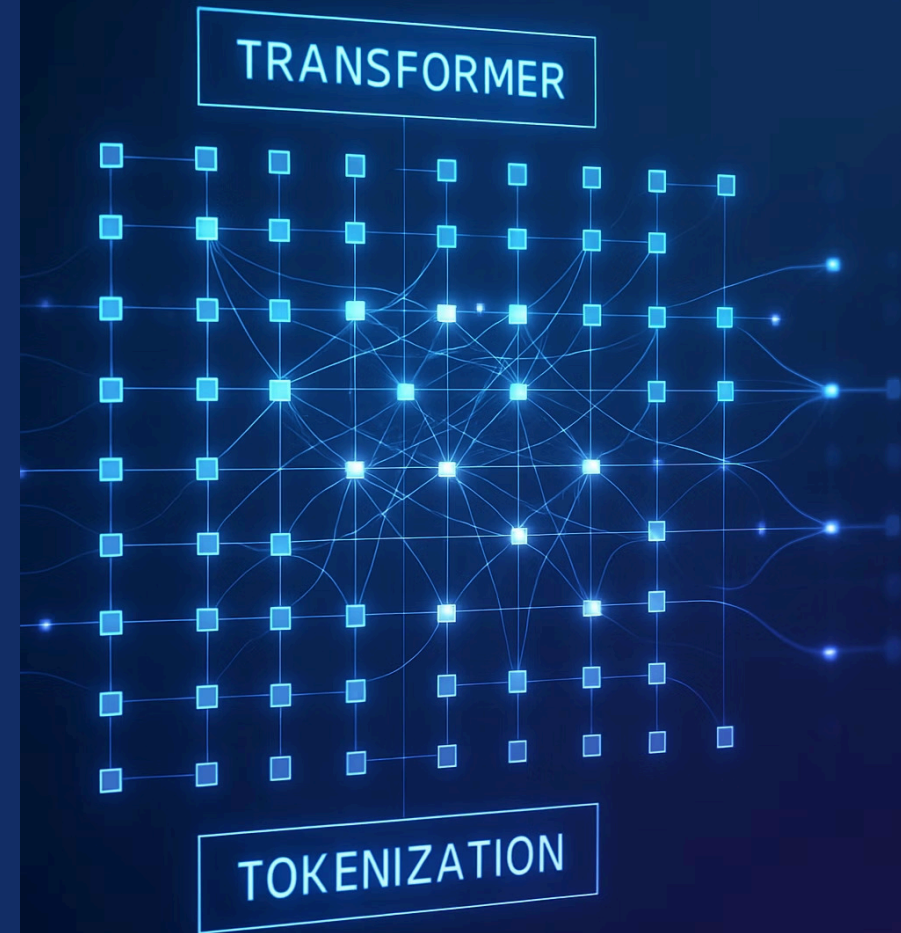


Transformers and Tokenizers in Large Language Models



by Eyal Ben-Zion



Introduction to Large Language Models

AI Paradigm Shift

LLMs reshape NLP landscape, propelling toward AGI

Core Components

Transformer architecture provides backbone, tokenizers interface with language

Synergistic Design

Performance depends on co-evolution of architecture and tokenization



The Transformer Revolution



Key Transformer Mechanisms



Scaled Dot-Product Attention

$$\text{Attention}(Q,K,V) = \text{softmax}(QK^T / \sqrt{d})V$$



Self-Attention

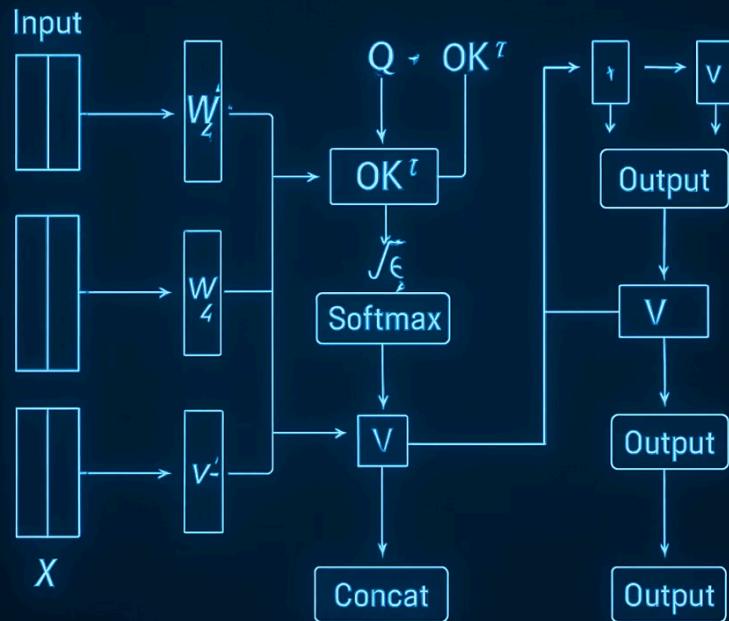
Captures global dependencies between tokens



Multi-Head Attention

Parallel attention functions capture diverse relationships

TRANSFORMER ATTENTION MECHANISM



Transformer Advantages

Parallelizability

Processes entire sequence concurrently

- Faster training times
- Efficient for long inputs
- Enables massive scaling

Computational Challenges

Quadratic complexity with sequence length

- $O(N^2)$ time and space complexity
- Bottleneck at scale
- Requires efficiency innovations

Efficiency Enhancements in LLMs



Efficiency Enhancement Strategies

Category	Strategy	Description
Pre-training	Data Curation	Optimize selection to reduce costs
Fine-tuning	Layer-wise Fine-tuning	Focus on higher layers for tasks
Model Design	Sparse Attention	Limit attention to subset of keys
Model Design	Mixture-of-Experts	Route through multiple sub-networks
Compression	Pruning & Quantization	Remove weights, reduce precision



Long-Context Challenges



Current Limitations

Pre-trained on short text, struggles with long contexts



Quadratic Complexity

$O(N^2)$ time/space limits input length



Optimization Approaches

Local, hierarchical, sparse attention mechanisms

Tokenization: Text to Computation

Critical Interface

Transforms raw text into discrete tokens for LLMs

Vocabulary Impact

Directly influences model's language processing abilities

Subword Approach

Balances character and word-level representations



Subword Tokenization Algorithms



BPE (17 citations)

Merges frequent character pairs iteratively



WordPiece (21 citations)

Uses longest-match-first tokenization strategy



SentencePiece (23 citations)

Language-independent tokenization approach



Unigram (5 citations)

Prunes vocabulary from a large initial set

Each algorithm offers different trade-offs in vocabulary management and language processing efficiency.

Tokenizer Integration in LLM Pipeline



Traditional Approach

Static, decoupled from model learning



Evolving Role

Moving toward dynamic adaptation



Adaptive Tokenizers

Refine vocabulary based on model perplexity



Future Direction

Integral, learning components of LLM system



Tokenization Challenges and Biases



Static Nature

Fixed, non-adaptive to model learning



Tokenization Bias

Different distributions over next byte prediction



Computational Language Bias

English 13x more efficient than some languages



OOV Problem

Challenges with specialized domains like code





Language Tokenization Efficiency Impact

1x

English

Baseline efficiency, optimal
context use

2x

French

Cost: \$4/student vs \$2 for
English

5x

Ukrainian

Cost: \$10/student, shorter
context

13.5x

Tamil

Cost: \$27/student, severely
limited context

Advanced Tokenization Strategies



Adaptive Tokenizers

Co-evolve with model learning patterns



Tokenizer-Free LLMs

T-Free and HAT models bypass tokenization



Specialized Tokenization

Domain-specific for code, multimodal inputs



Bias Mitigation

Expanded context windows, targeted pre-training



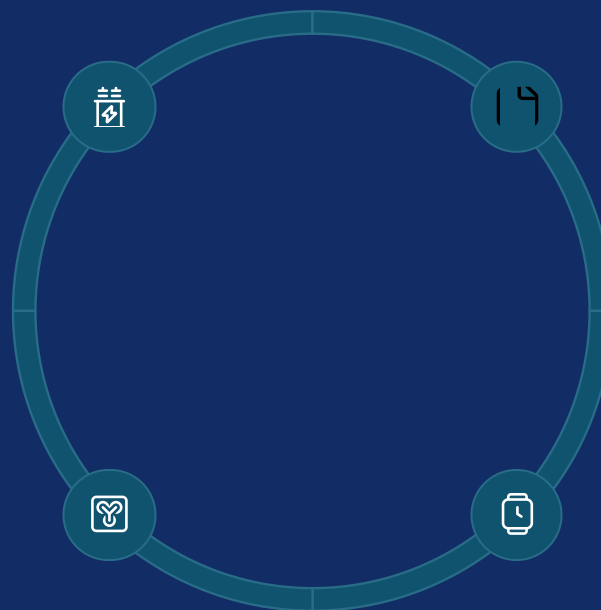
Conclusions and Future Directions

Transformer Impact

Revolutionized sequence modeling with parallel processing

Holistic Design

Balancing performance, sustainability, and linguistic fairness



Tokenizer Evolution

Moving from static to adaptive, learnable approaches

Equity Challenges

Addressing computational language bias critical for access

Short podcast in Hebrew about the lecture

Transformer architecture and tokenization strategies. The Transformer serves as the architectural backbone for the majority of contemporary LLMs, while tokenizers act as the critical interface, converting raw, unstructured natural language into a discrete, computationally amenable format.¹ The effectiveness of an LLM is profoundly influenced by its tokenizer, as this preprocessing step directly determines the model's vocabulary and, consequently, its ability to comprehend and produce language.⁵

The success observed in LLMs is a direct consequence of the co-dependent evolution and integration of the Transformer architecture and advanced tokenization strategies. The remarkable parallelizability of the Transformer architecture, a key advantage over its predecessors, is intrinsically linked to how efficiently the tokenizer segments input sequences. For instance, the ability of Transformers to process entire token sequences simultaneously¹⁰ relies heavily on the quality and efficiency of the tokenized input. Conversely, contemporary research is actively engineering advanced tokenization methods for seamless integration with LLM architectures, indicating a co-evolutionary design philosophy. This suggests that achieving optimal LLM performance is not merely a matter of developing a powerful Transformer or an efficient tokenizer in isolation, but rather hinges on their synergistic design and

 Google Docs



Transformers, Tokenizers in LLM Review

A Comprehensive Review of Transformers and Tokenizers in Large Language Models for Graduate Engineers 1. Introduction to Large Language Models (LLMs)...

[שנאים ואסימונים במודלי שפה גדולים.wav](https://wav.גדולים.שפה)

