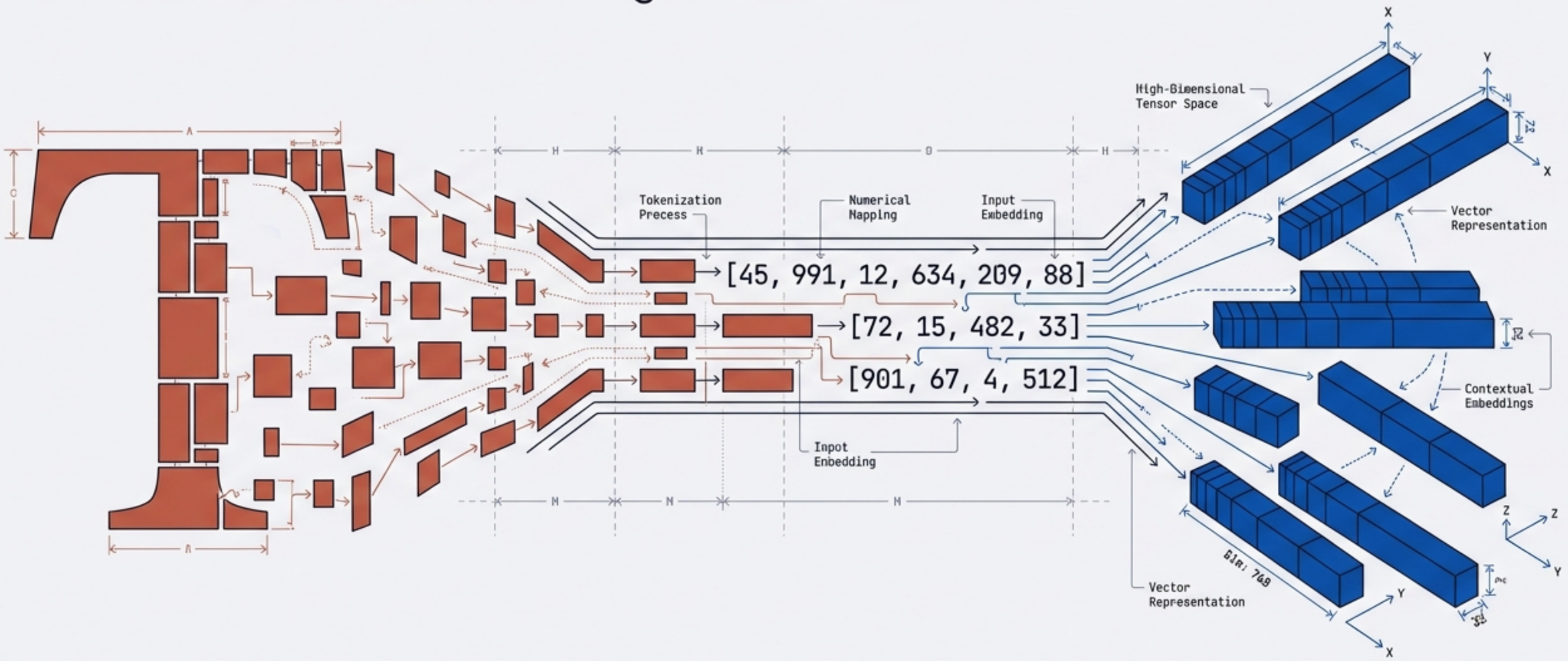
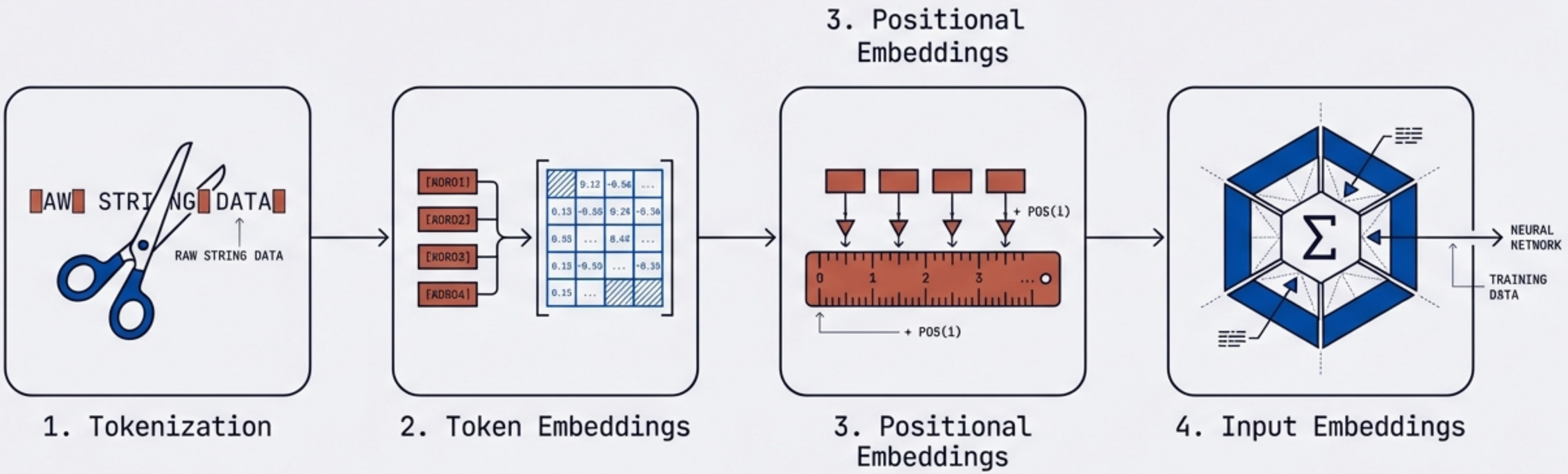
The LLM Data Metamorphosis

From Raw Unstructured Text to High-Dimensional Tensors



Architecture: GPT-2 / PyTorch Implementation

The Preprocessing Pipeline

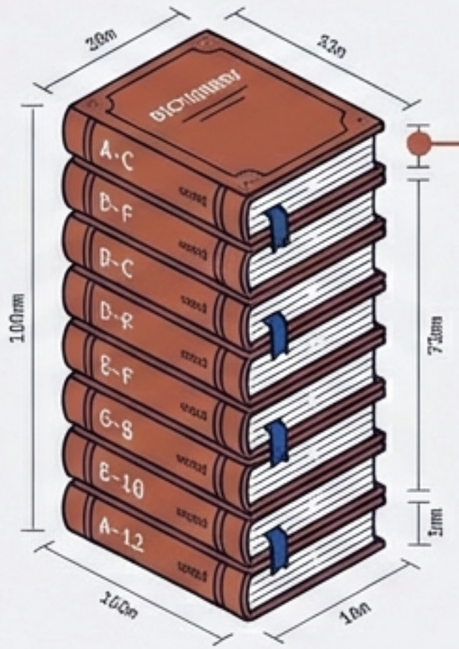


Large Language Models cannot process PDFs or raw strings. Data preprocessing is the fundamental translation layer that converts human sentences into the numerical format required for neural network training.

The Tokenization Dilemma

Word-Based

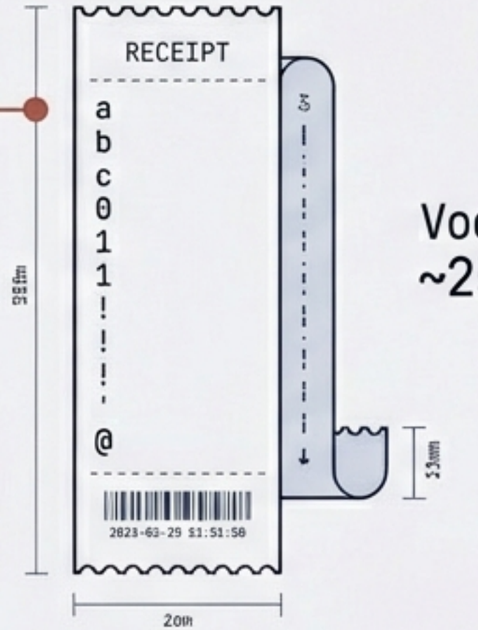
Vocabulary: ~600k–1M words



Fails on unknown words. Loses root relationships (Boy vs Boys).

Character-Based

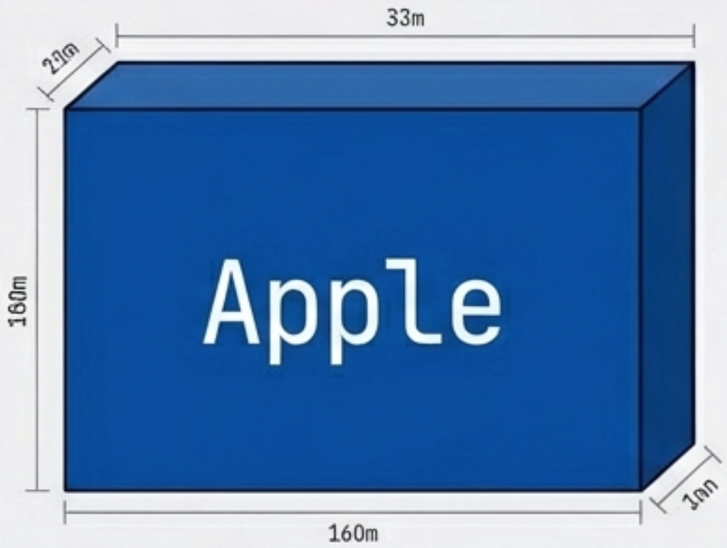
Loses semantic meaning. Sequences become impractically long.



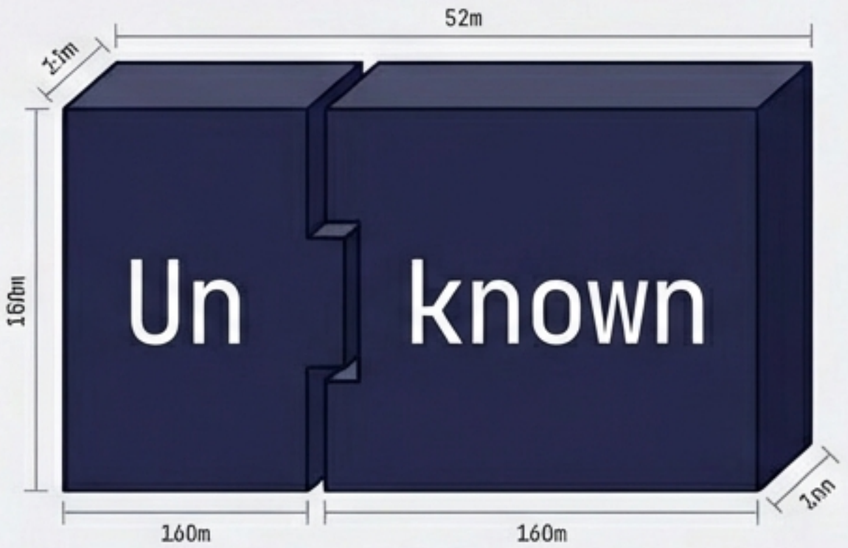
Vocabulary: ~256 chars

The Goal: Balance efficiency with meaning.

The Solution: Subword Tokenization & BPE



Common Word = Whole Token



Rare Word = Subword Split

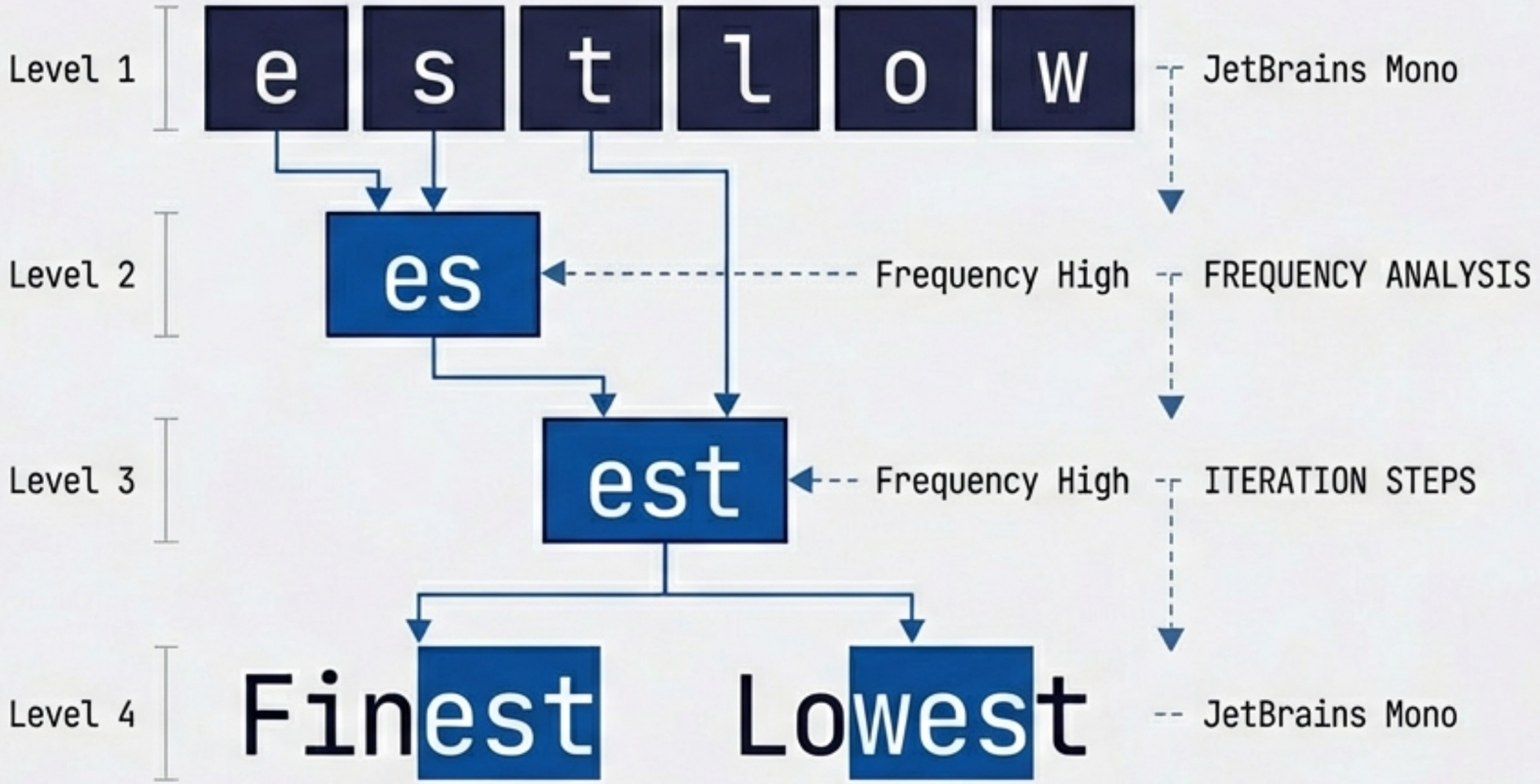
Byte Pair Encoding (BPE) keeps frequent words whole but breaks rare words into meaningful sub-units.

Optimizes vocabulary size (~50,000 for GPT-2) while retaining semantic roots.

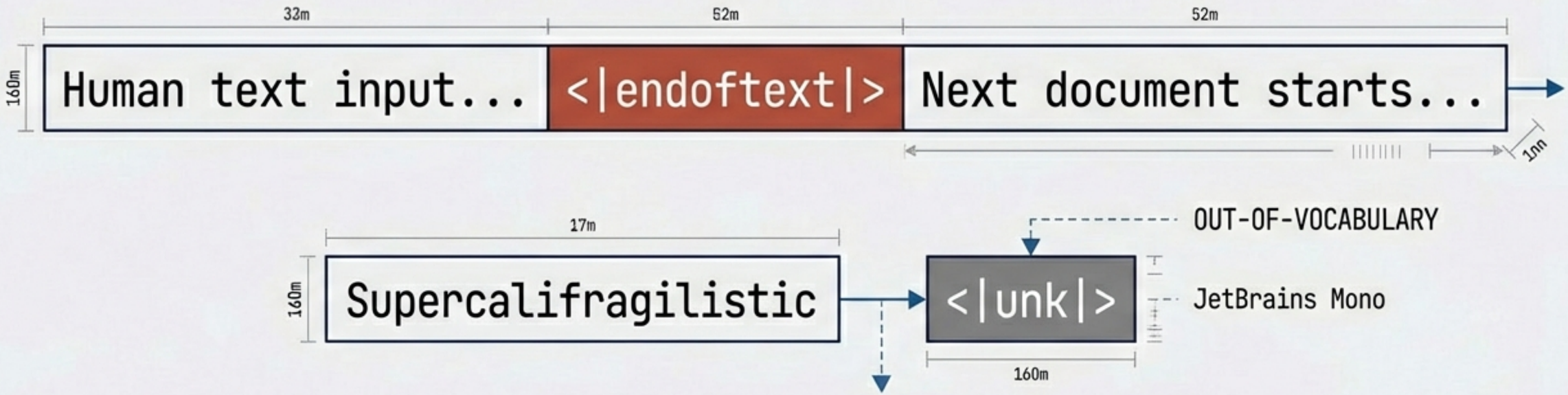
The Goal: Balance efficiency with meaning.

The BPE Algorithm: Iterative Merging

The algorithm statistically identifies and merges the most frequent adjacent characters to build a vocabulary.

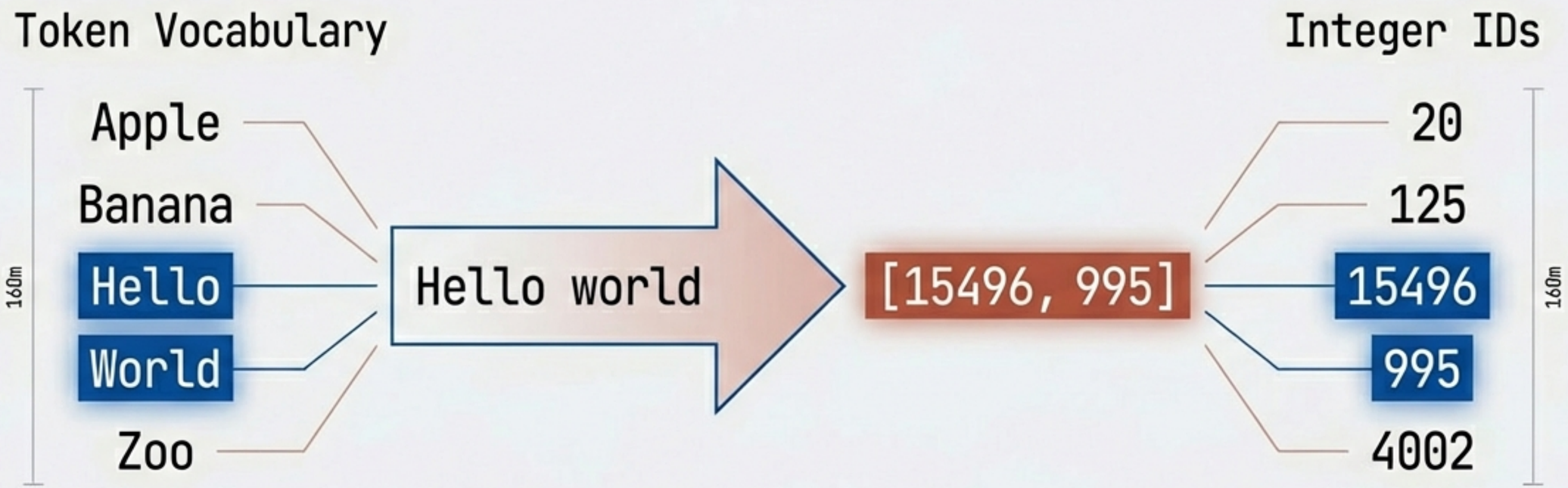


Handling Context & Boundaries



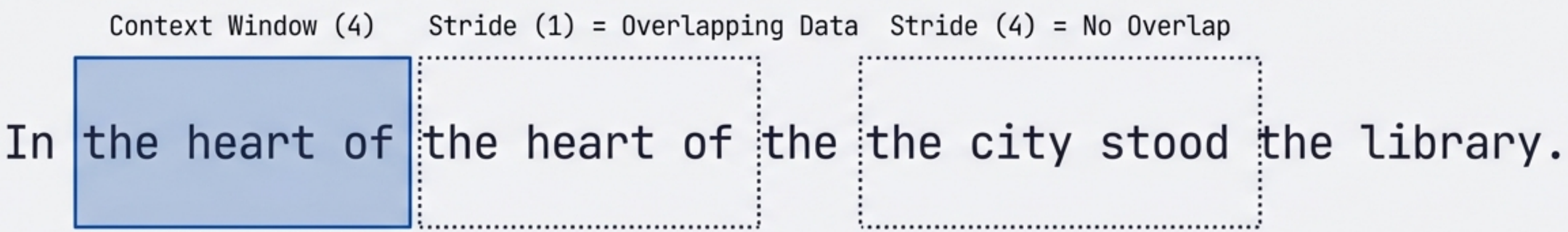
- `<|endoftext|>`: Delineates unrelated documents (e.g., News Article vs. Reddit Post).
- `<|unk|>`: Fallback for data outside the vocabulary (Rare in modern BPE).

From Tokens to IDs



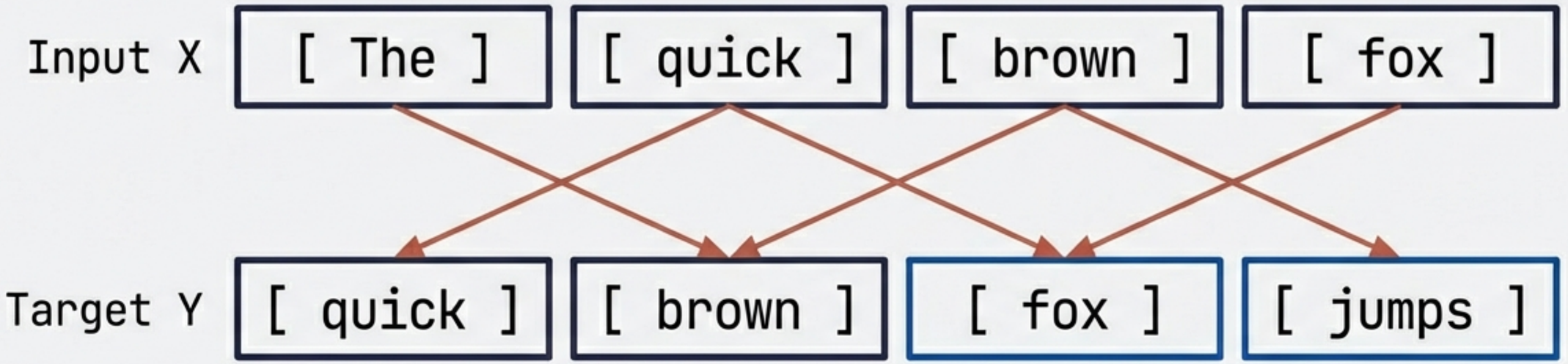
The machine does not see text; it sees a sequence of integers.
The vocabulary is a static dictionary lookup.

The Logistics: Context & Stride



- **Context Size:** Max tokens processed at once (e.g., 256, 1024).
- **Stride:** The step size for the sliding window moving through the corpus.

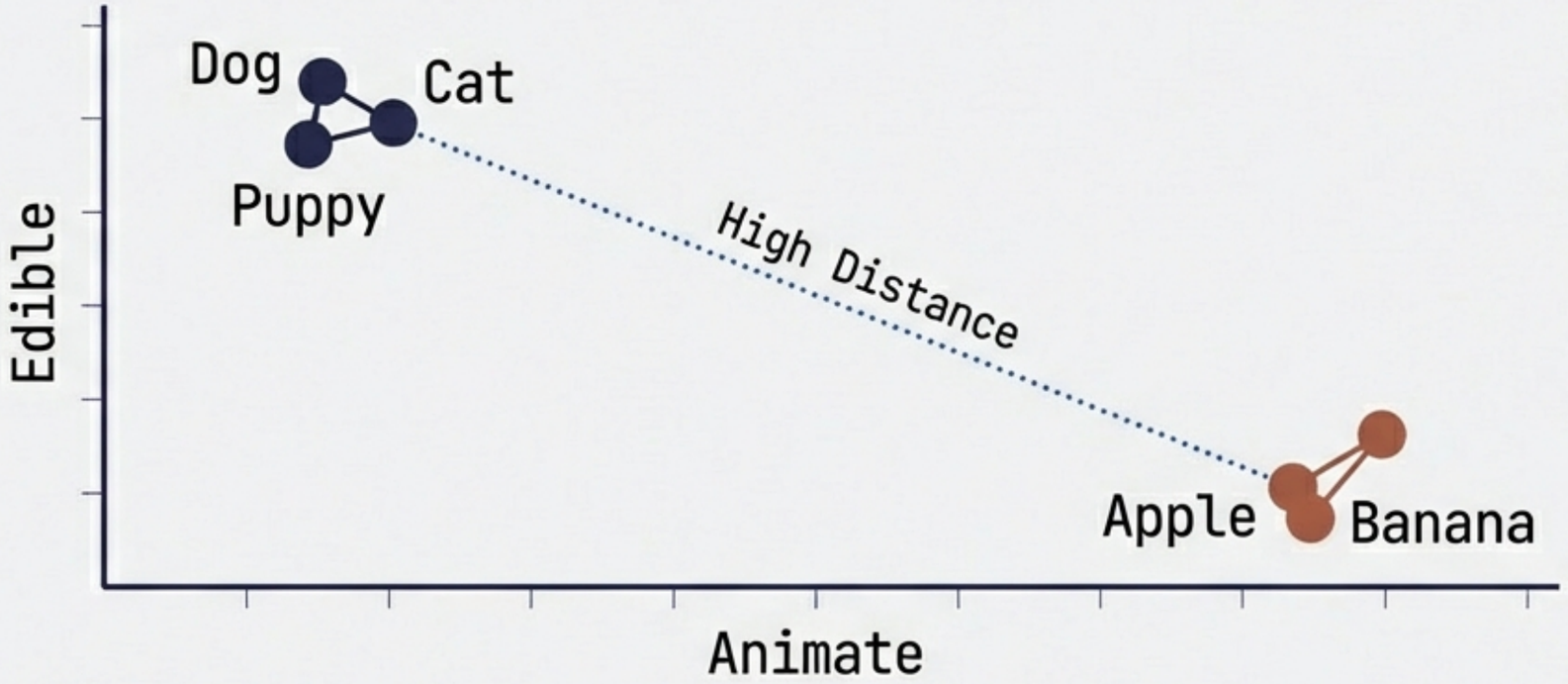
The Prediction Task



Input vs. Target (Shifted by 1).

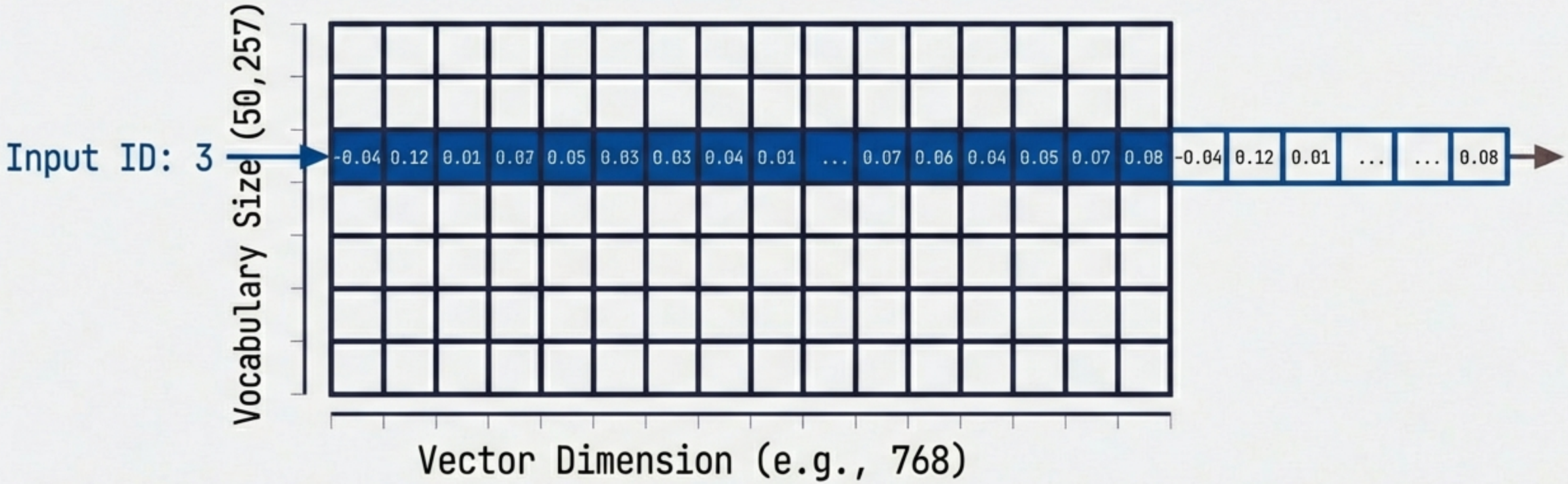
- **Insight:** A single context window of 4 tokens contains 4 parallel prediction tasks.

Token Embeddings: Injecting Meaning



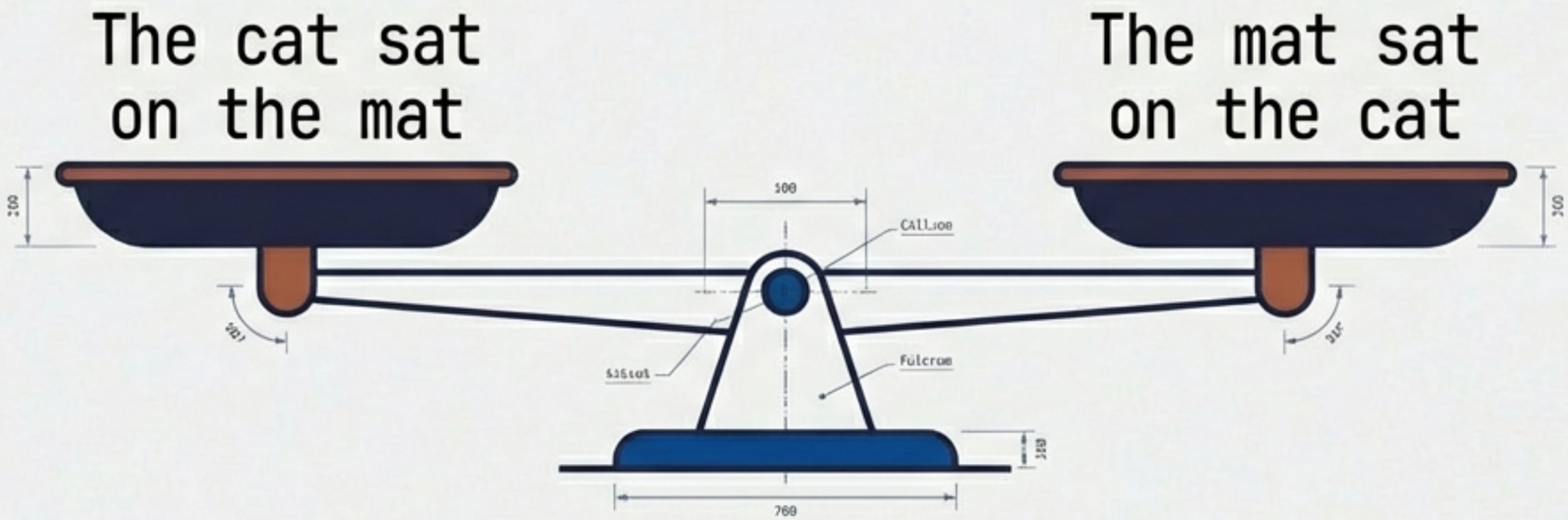
Integer IDs are arbitrary. Embeddings map tokens to geometric space where distance = semantic similarity.

The Embedding Layer



The Embedding Layer is a massive lookup table. It is a learnable weight matrix initialized randomly and optimized during training.

The Problem of Position



Sum(Embedding A) == Sum(Embedding B)

Without positional info, the model is permutation invariant.
It sees a ‘bag of words’ with no concept of order.

Absolute Positional Embeddings

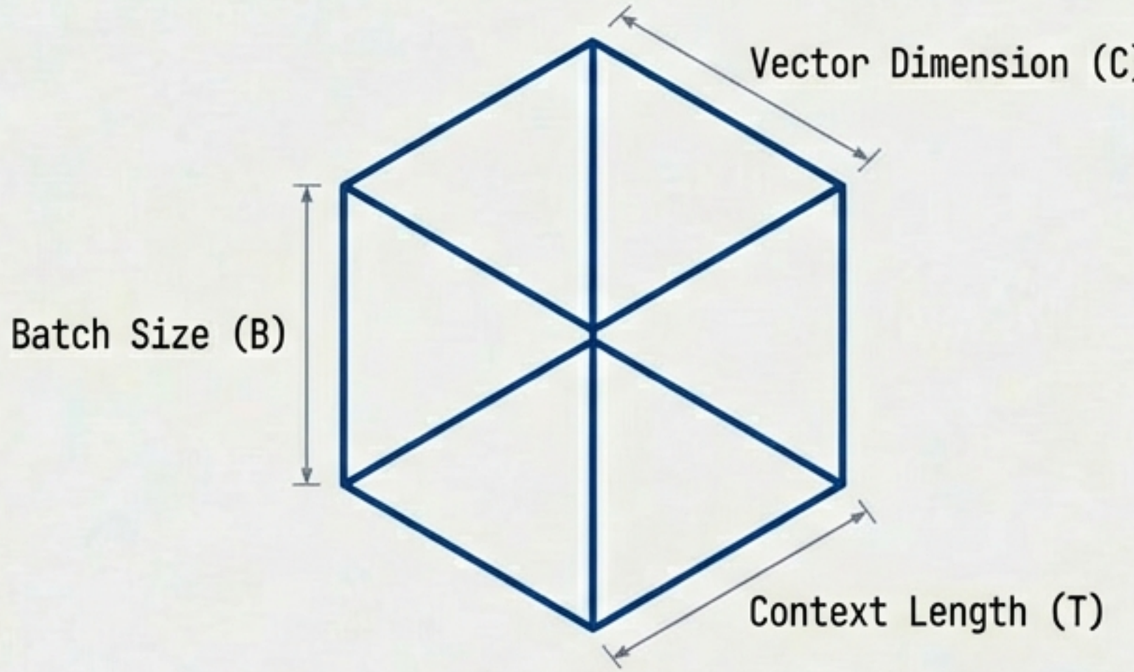
Token Vector
(Meaning)

Position Vector
(Index)

Input
Embedding

We create a unique vector for every position (0, 1, 2...)
and add it element-wise to the token vector.

The Final Assembly: Input Embeddings



The B × T × C Tensor

This tensor is the fuel for the Transformer. It fuses word identity, semantic meaning, and positional context into a single numerical object.

The Pipeline Recap

- **Tokenization:** BPE balances vocabulary size and meaning.
- **Indexing:** Convert text to integers via learned vocabulary.
- **Embeddings:** Convert IDs to vectors for semantic relationships.
- **Positioning:** Add positional vectors to encode order.

A strongly performing Large Language Model begins with a robust Data Processing Pipeline.