Byte Pair Encoding (BPE) Tokenization

The Input Pipeline That Determines Everything Downstream

Byte Pair Encoding is the subword tokenization algorithm used by GPT-2, GPT-4, LLaMA, Mistral, and virtually every modern LLM. Starting from a base vocabulary of 256 byte tokens, BPE iteratively merges the most frequent adjacent pairs to build a vocabulary of subword units.

This demo covers:
1. Training process visualization: merge rules learned step by step
2. Compression ratio analysis: characters per token vs vocab size
3. Tokenization examples: English, code, Unicode, emoji
4. Vocabulary size tradeoff: sequence length, memory, compute
5. Byte fallback & special tokens: handling any UTF-8 input
6. Inference impact: KV cache and attention cost differences

Training corpus: 24 documents
Random seed: 42
Number of visualizations: 6
Examples: 6

Mathematical Foundation

BPE Training Algorithm

Given corpus C and target vocabulary size $|V|$:

$V_0 = \{b_i : i \in [0, 255]\}$ (base byte vocabulary)

For $k = 1, \ldots, |V| - 256$:

$(a^*, b^*) = \arg\max_{(a,b)} \sum_{w \in C} \text{count}((a, b) \text{ in } w) \cdot \text{freq}(w)$

$V_k = V_{k-1} \cup \{a^* \| b^*\}$, replace all (a^*, b^*) with $a^* \| b^*$ in C

Encoding (Applying Merge Rules)

Given text t and ordered merge rules $M = [(a_1, b_1), \ldots, (a_m, b_m)]$:

$\text{tokens}_0 = [c_1, c_2, \ldots, c_n]$ where c_i are byte tokens of t

For each $(a_k, b_k) \in M$: replace all adjacent (a_k, b_k) with $a_k \| b_k$

$\text{IDs} = [\text{vocab[tok] for tok in final tokens}]$

Inference Cost Analysis

Let n = sequence length (tokens), L = layers, h = heads, d_h = head dim:

$\text{Attention FLOPs } = 2 \cdot L \cdot h \cdot n^2 \cdot d_h \quad \Rightarrow \quad O(n^2)$

$\text{KV cache memory } = 2 \cdot L \cdot h \cdot n \cdot d_h \cdot b \text{ bytes} \quad \Rightarrow \quad O(n)$

$\text{Embedding memory } = |V| \cdot d_{\text{model}} \cdot b \text{ bytes} \quad \Rightarrow \quad O(|V|)$

Compression Ratio

$\text{CPT} = \frac{|\text{text}|}{|\text{encode(text)}|}$ (characters per token, higher = better)

$\text{CR} = \frac{|\text{text}|_{\text{bytes}}}{|\text{encode(text)}|}$ (compression ratio)

Vocabulary Size Tradeoff

Smaller $|V|$: longer $n \Rightarrow$ attention $O(n^2) \uparrow$, KV cache $O(n) \uparrow$, embedding $O(|V|) \downarrow$

Larger $|V|$: shorter $n \Rightarrow$ attention $O(n^2) \downarrow$, KV cache $O(n) \downarrow$, embedding $O(|V|) \uparrow$

Summary of Findings

1. Training Process: BPE greedily merges the most frequent adjacent byte pairs. The best-pair frequency decays as obvious patterns are consumed. Later merges create progressively longer tokens by combining previously merged tokens.

2. Compression: Characters per token (CPT) improves with vocabulary size, with diminishing returns. English text trained on English corpus achieves best CPT. Random/unseen text compresses poorly (~1.0 CPT, byte-level fallback).

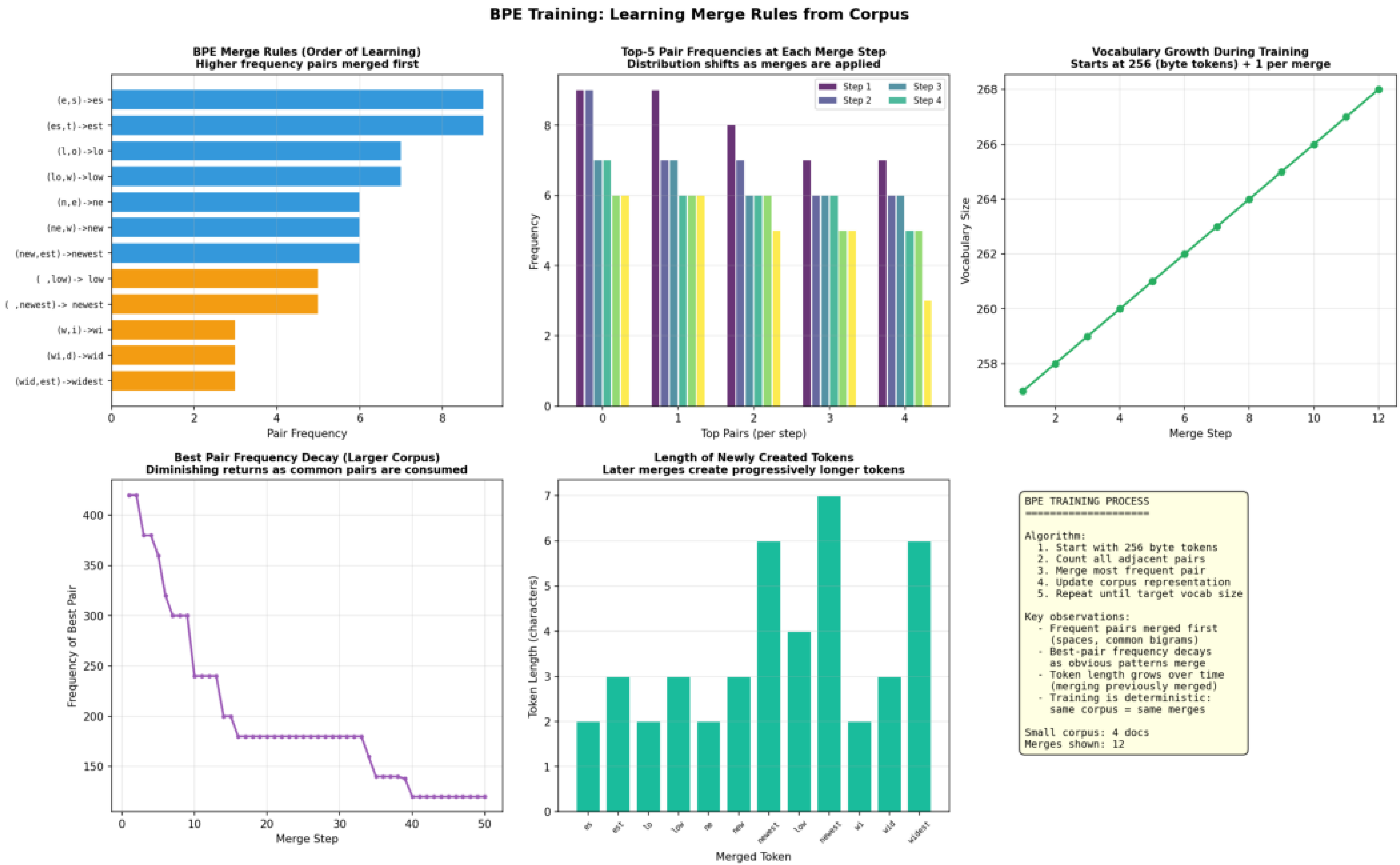
3. Tokenization: English text compresses well (high CPT). Code is moderate. CJK, Arabic, and emoji fall back to byte-level encoding (multiple tokens per character) since the training corpus is English-heavy.

4. Vocab Size Tradeoff: Larger vocabulary produces shorter sequences, reducing attention compute (O(n^2)) and KV cache memory (O(n)), but increases embedding table size (O(|V|)). Sweet spot: 32K-100K for production LLMs.

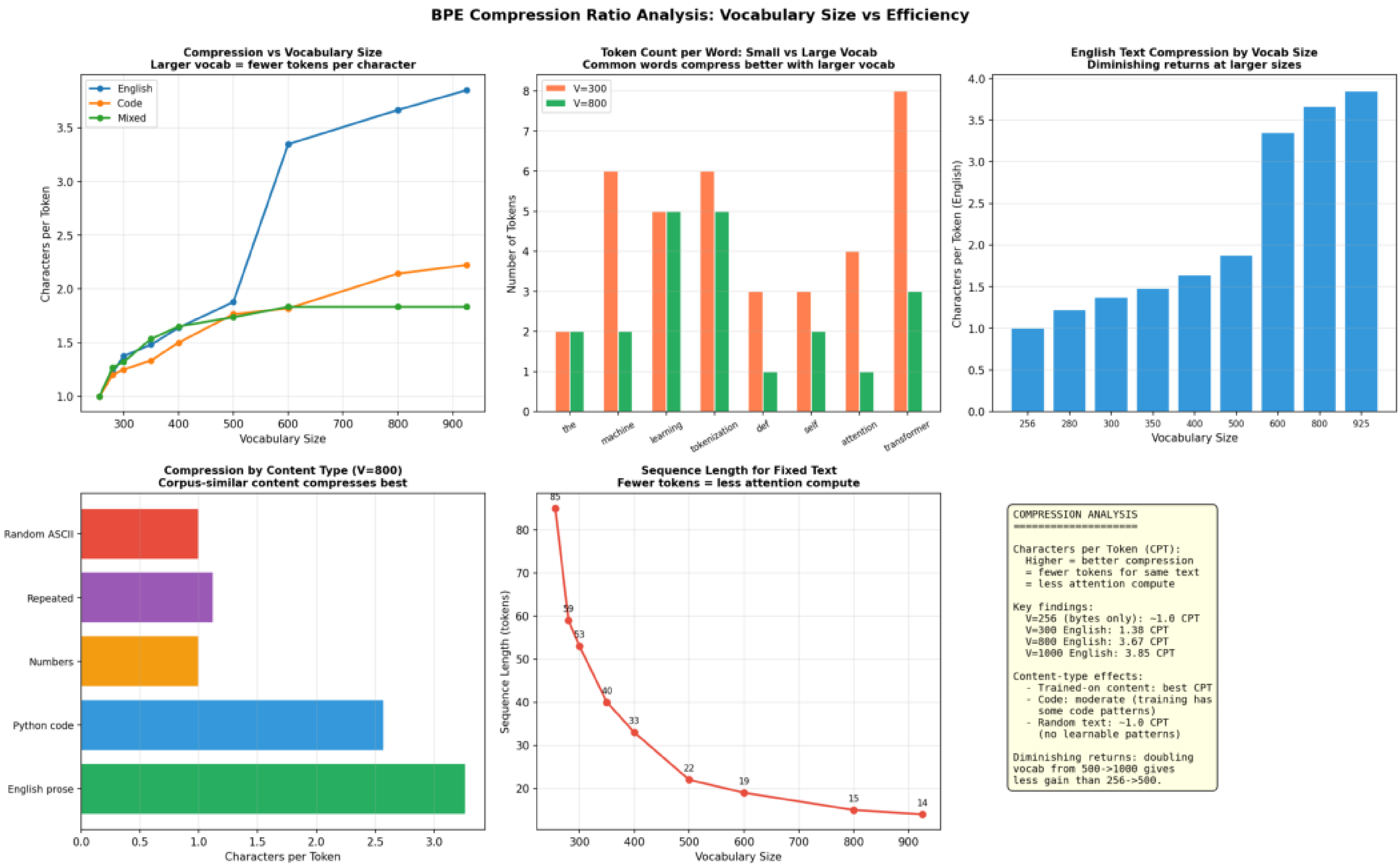
5. Byte Fallback: The 256 base byte tokens guarantee encoding of ANY UTF-8 input. No <unk> token needed. Special tokens (e.g., <|endoftext|>) are protected from BPE splitting and always encode to a single token ID. Roundtrip correctness is guaranteed: decode(encode(text)) == text for all valid UTF-8.

6. Inference Impact: Better tokenization directly reduces inference cost. Shorter sequences mean less attention compute, smaller KV cache, and more concurrent requests per GPU. At batch scale, the memory savings from even modest CPT improvements translate to significant GPU memory freed for serving.

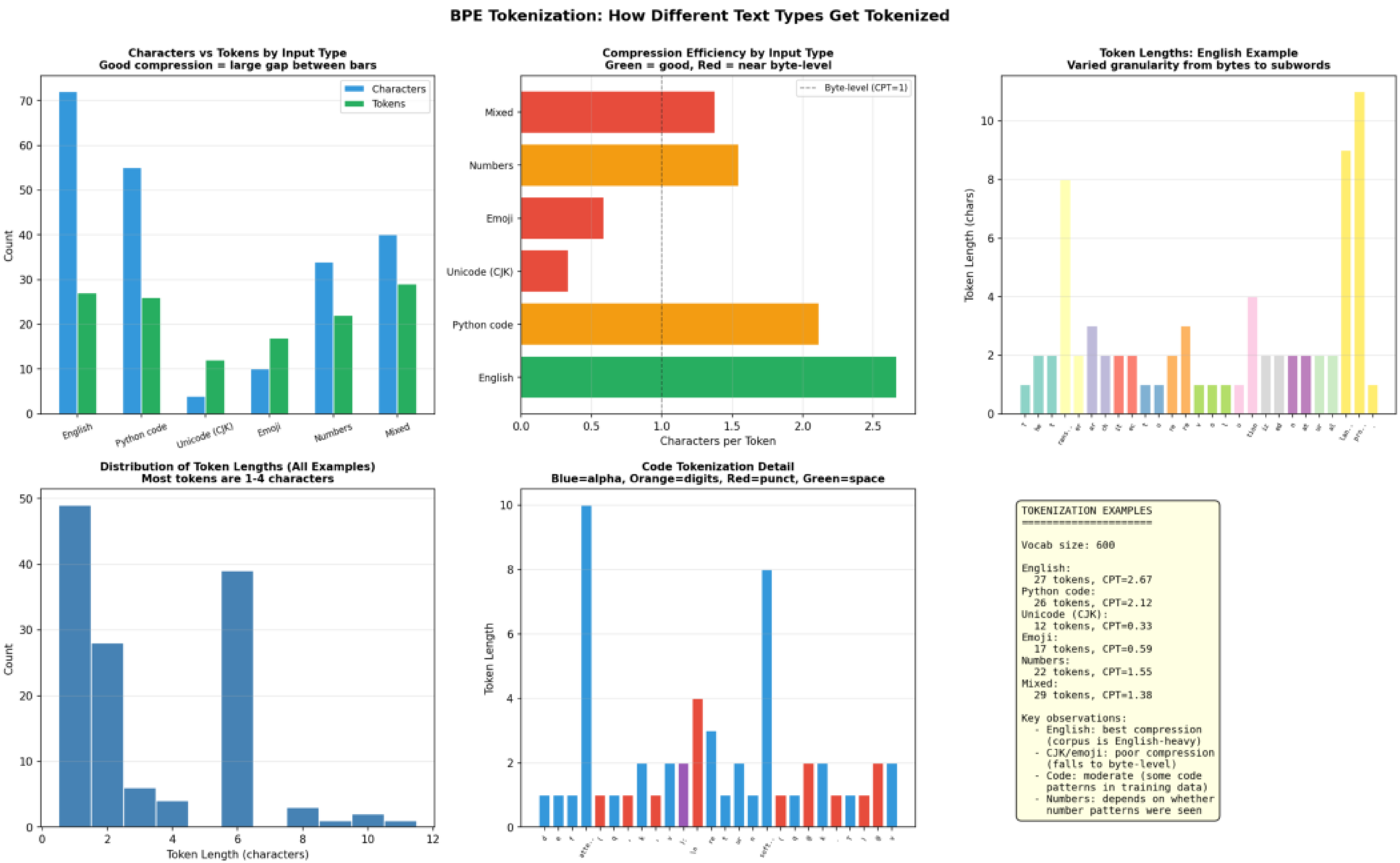
Example 1: BPE Training Process



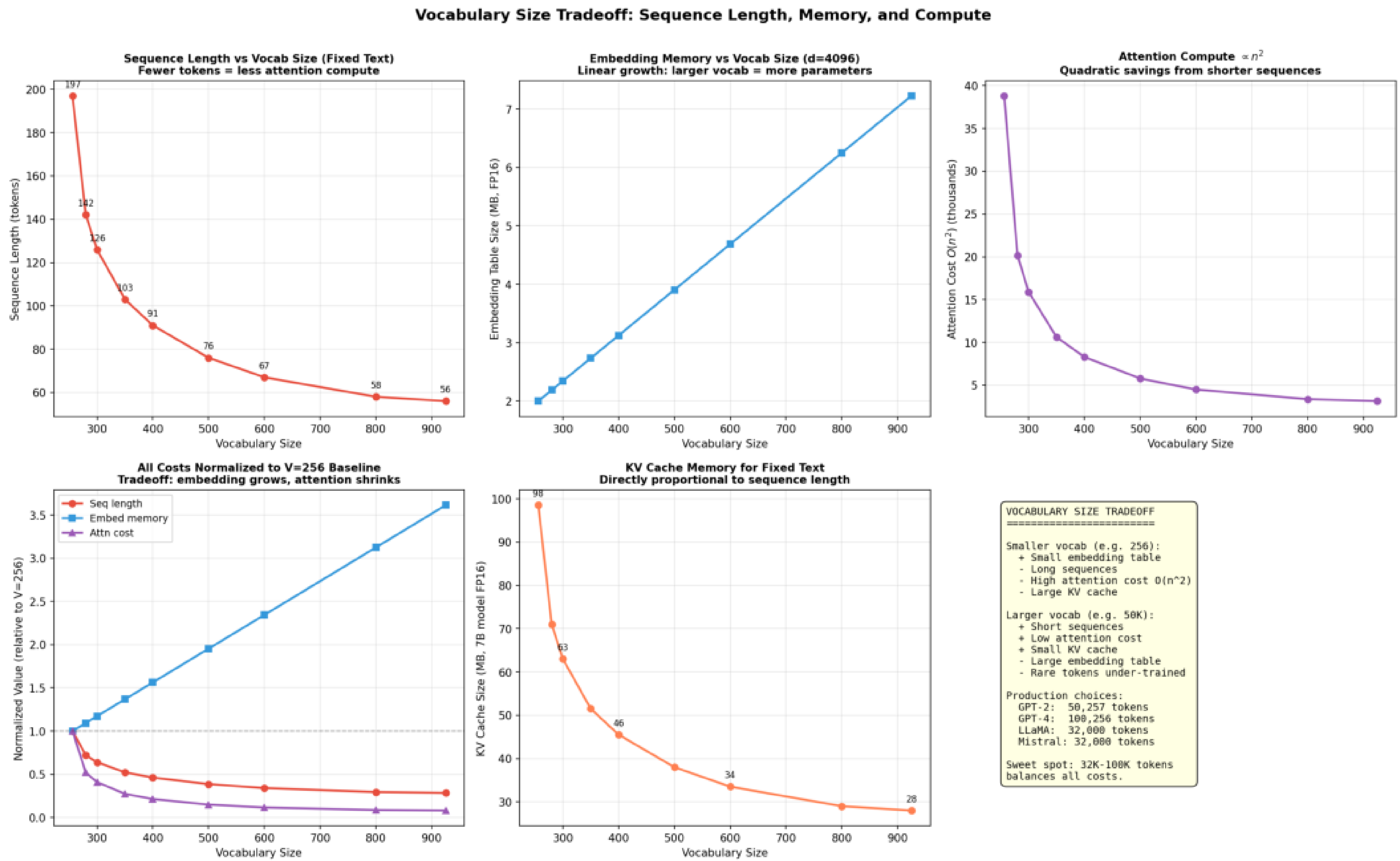
Example 2: Compression Ratio Analysis



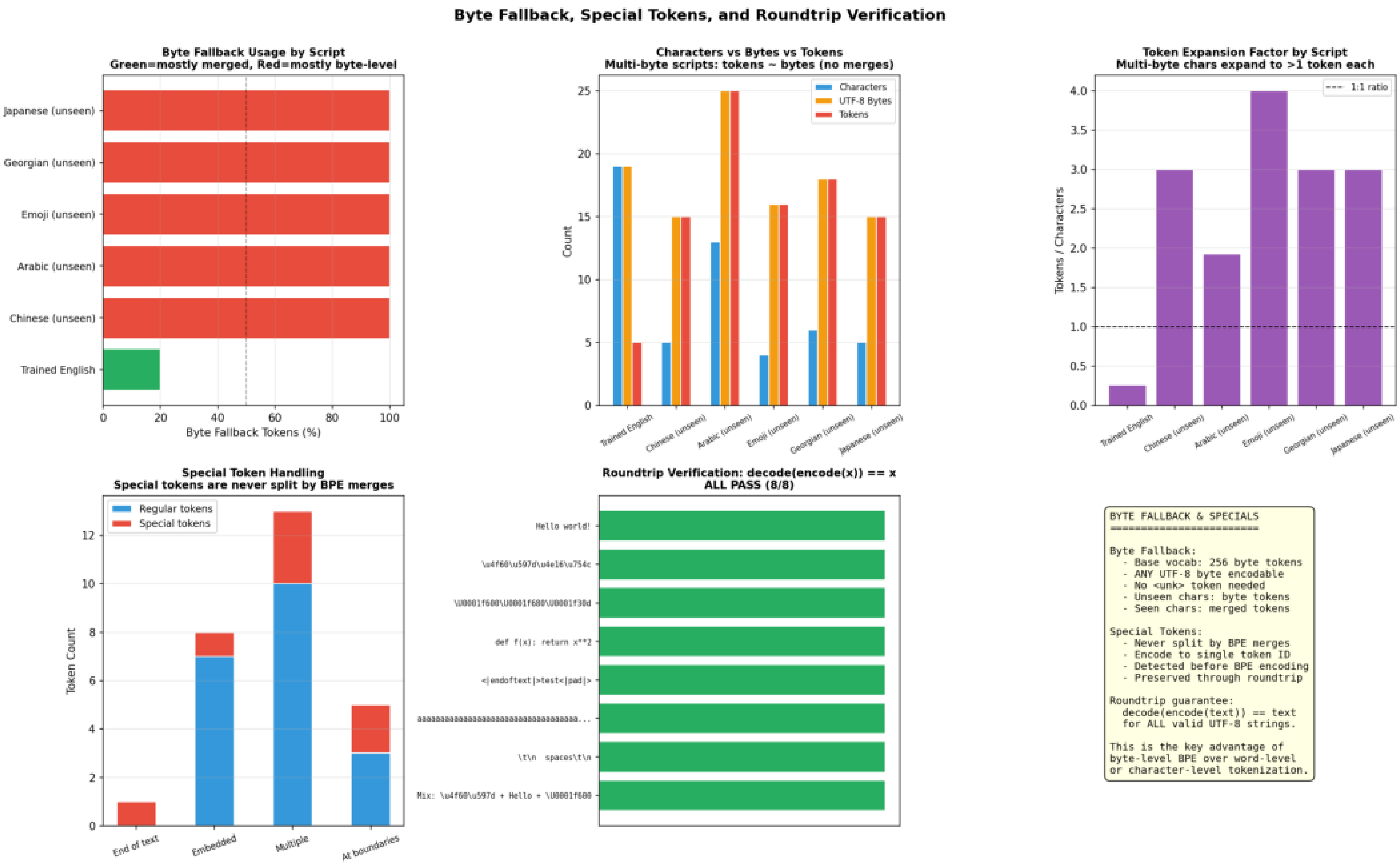
Example 3: Tokenization Examples



Example 4: Vocabulary Size Tradeoff



Example 5: Byte Fallback & Special Tokens



Example 6: Inference Impact Analysis