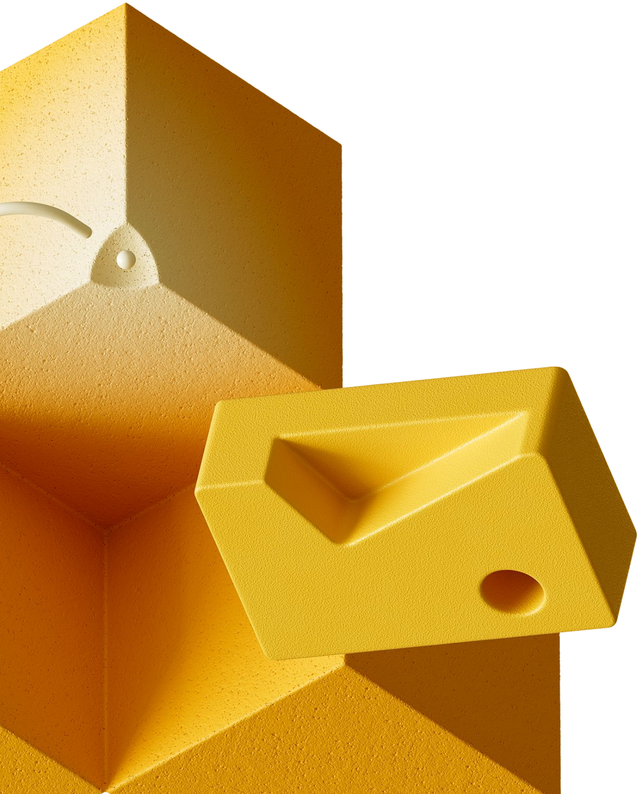


Training LLMs over Neurally Compressed Text

Brian Lester, Jaehoon Lee, Alex Alemi,
Jeffrey Pennington, Adam Roberts,
Jascha Sohl-Dickstein, Noah Constant

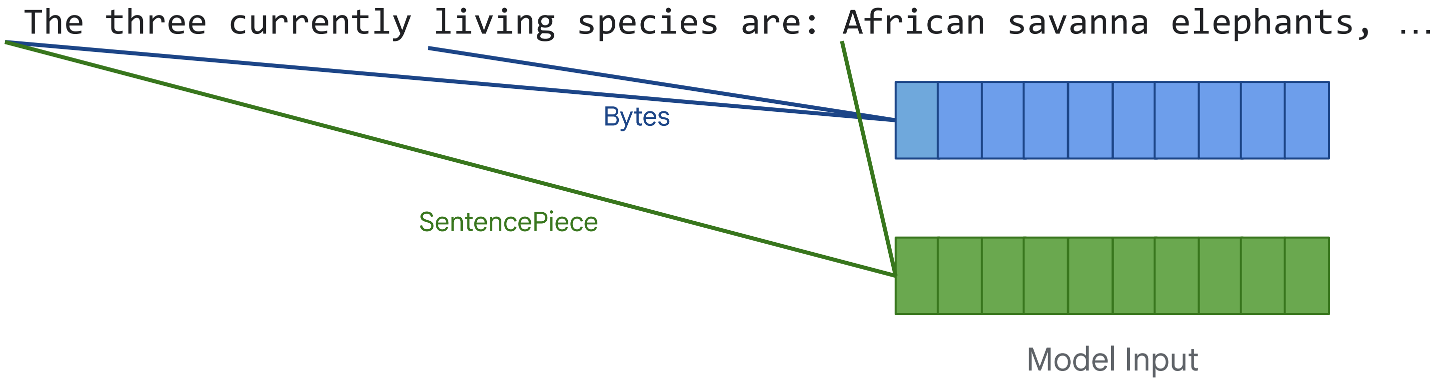
2025/04/27

Published in TMLR; Presented at ICLR 2025

01 Motivation

Why Compressed Text?

- "See" more raw text over the course of training
- "See" more text in your context window
- Shorter sequence lengths
- Tokenizers are compressors
  - BPE was invented as a compression algorithm



Beyond Current Tokenizers

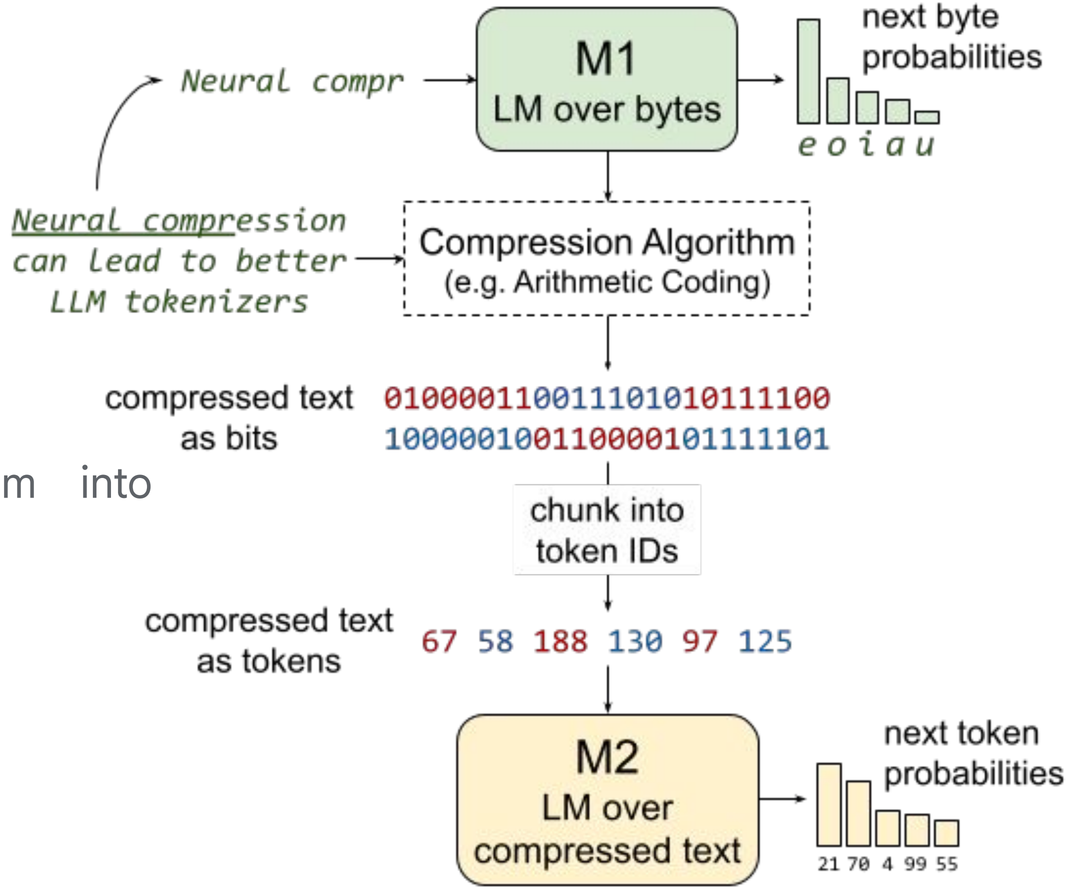
- Can we increase the compression rate?
  - Using LLMs in the compression
- Can the text in a token be more “visible” to the model

02

Setup

Training over Compressed Data

- Train a small model to predict the next byte
- Use the model's probabilities in the compression algorithm
- Segment the compressed bitstream into tokens
- Train a larger model on the compressed output



03 Results

Baselines

Baselines:

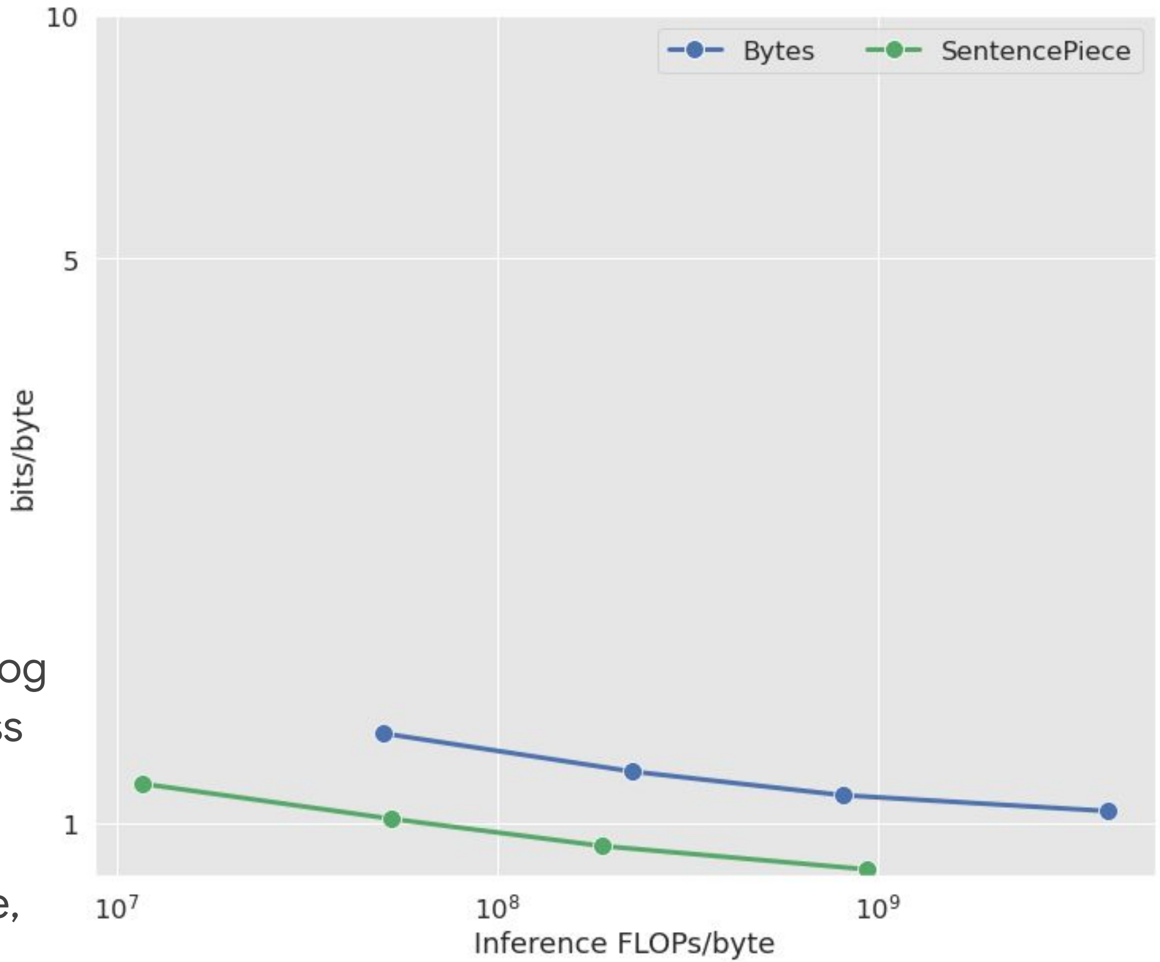
- Bytes
- SentencePiece

Model Sizes:

- 25m
- 113m
- 403m
- 2b

Bits/Byte: Normalized Negative Log Likelihood loss to compare across tokenizations (↓)

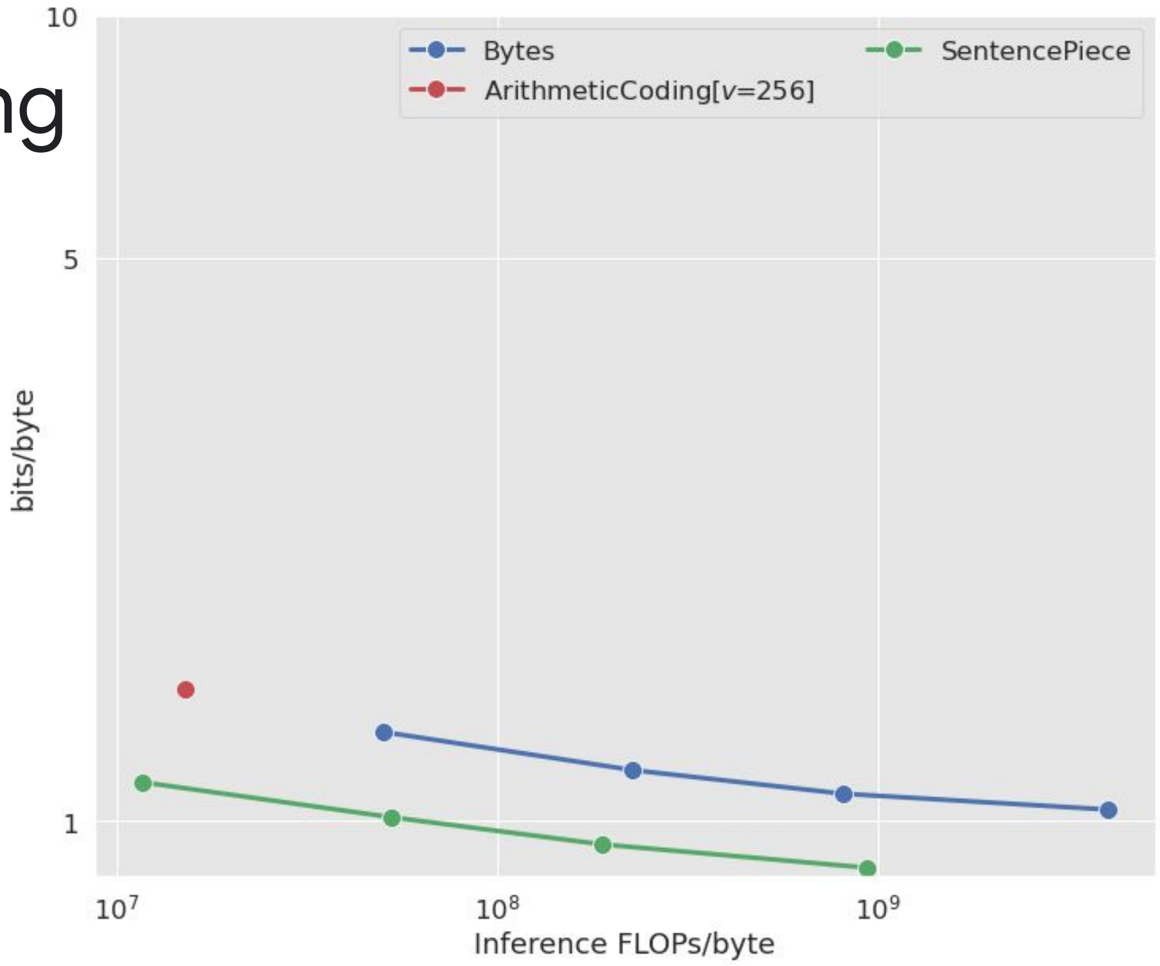
FLOPS/byte: Number of flops required to produce a single byte, based on compression ratio (⟵)



Arithmetic Coding

- Compress the whole sequence into a single example.
- Run AC compression based on M1 logits

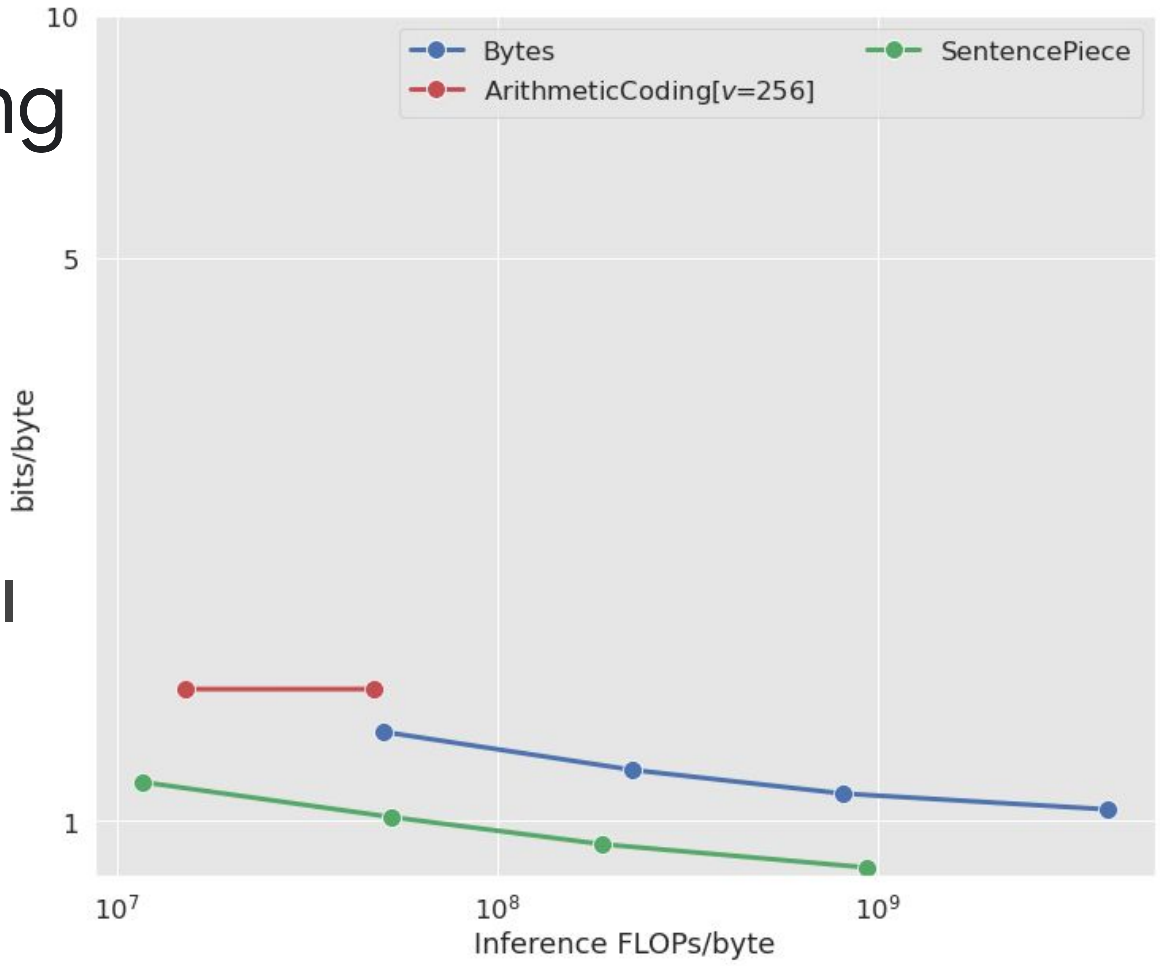
25m parameter model



Arithmetic Coding

- Compress the whole sequence into a single example.
- Run AC compression based on M1 logits

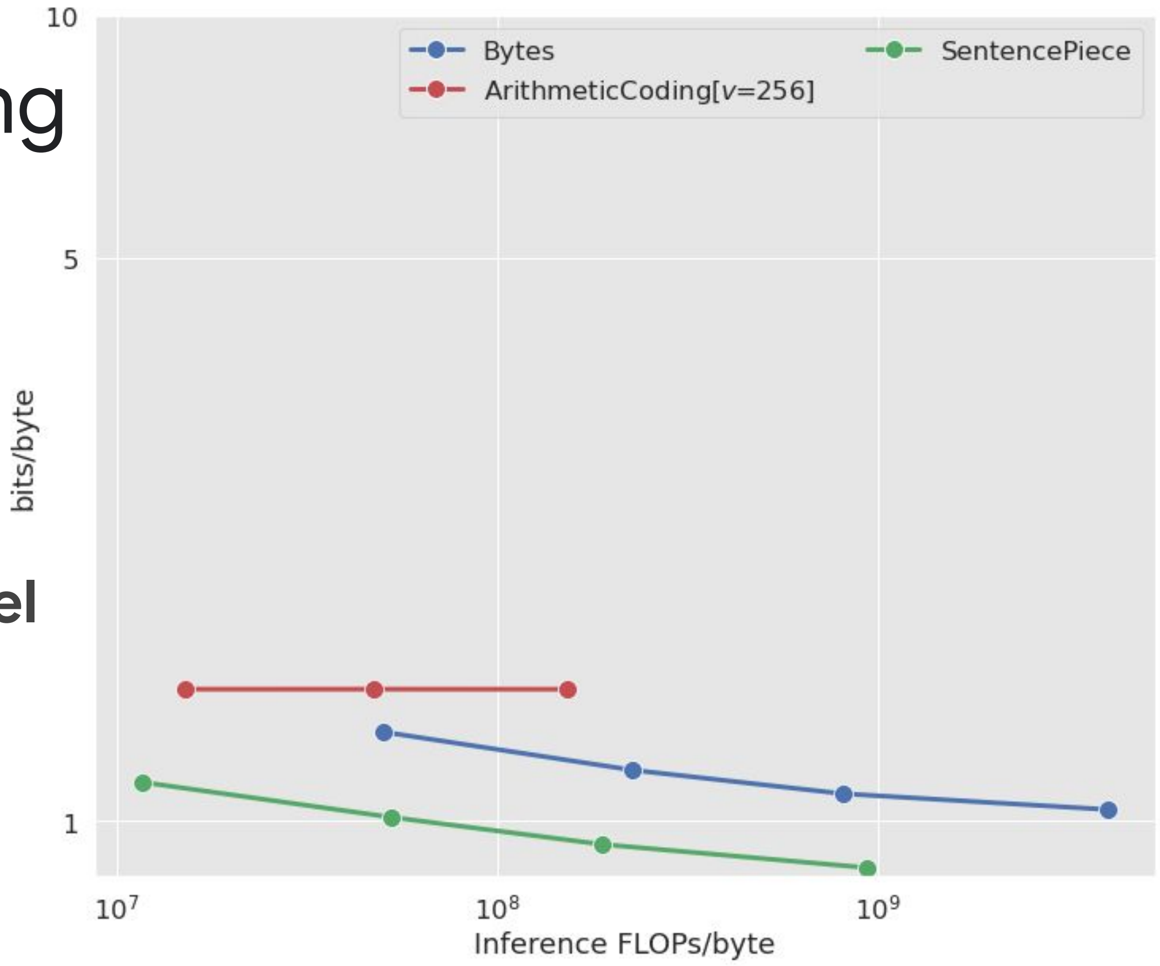
113m parameter model



Arithmetic Coding

- Compress the whole sequence into a single example.
- Run AC compression based on M1 logits

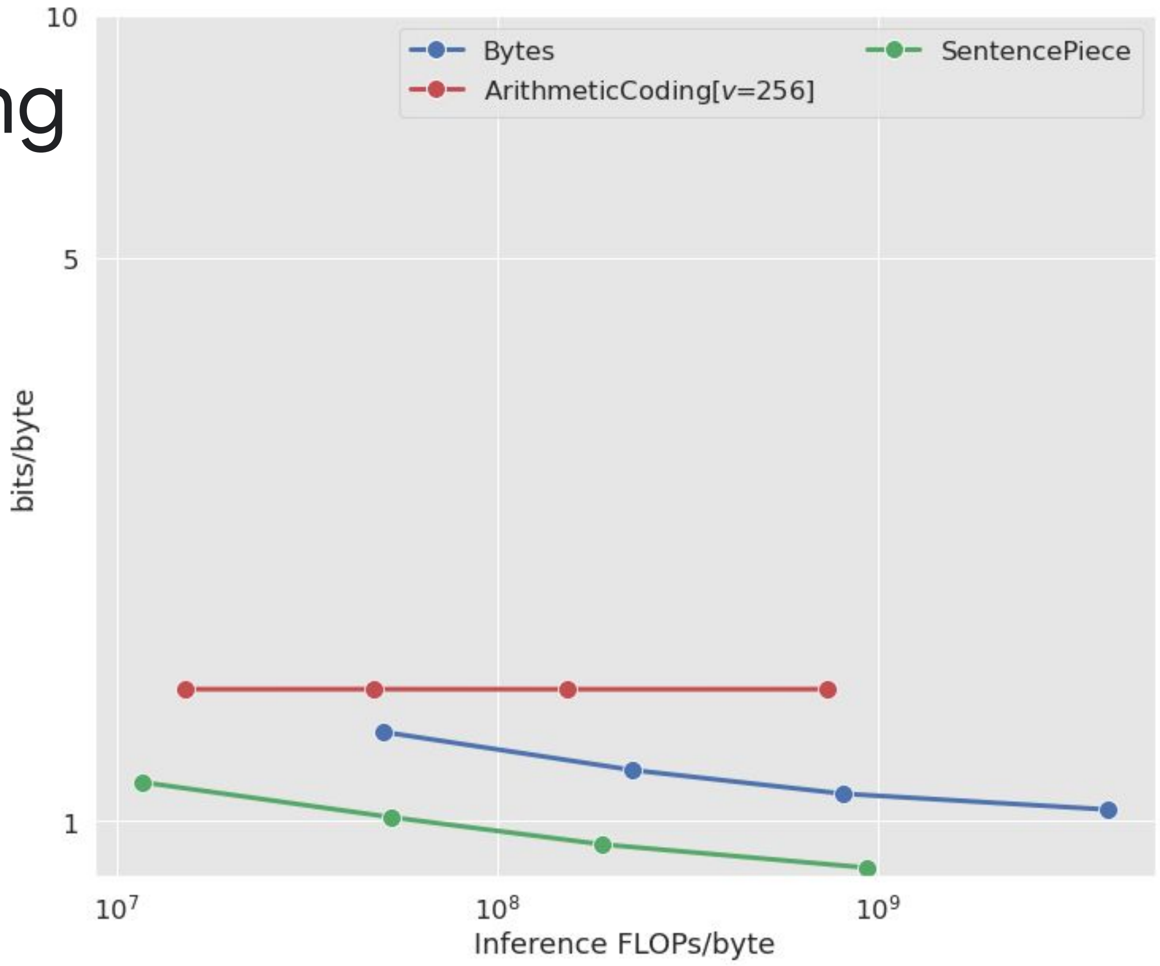
403m parameter model



Arithmetic Coding

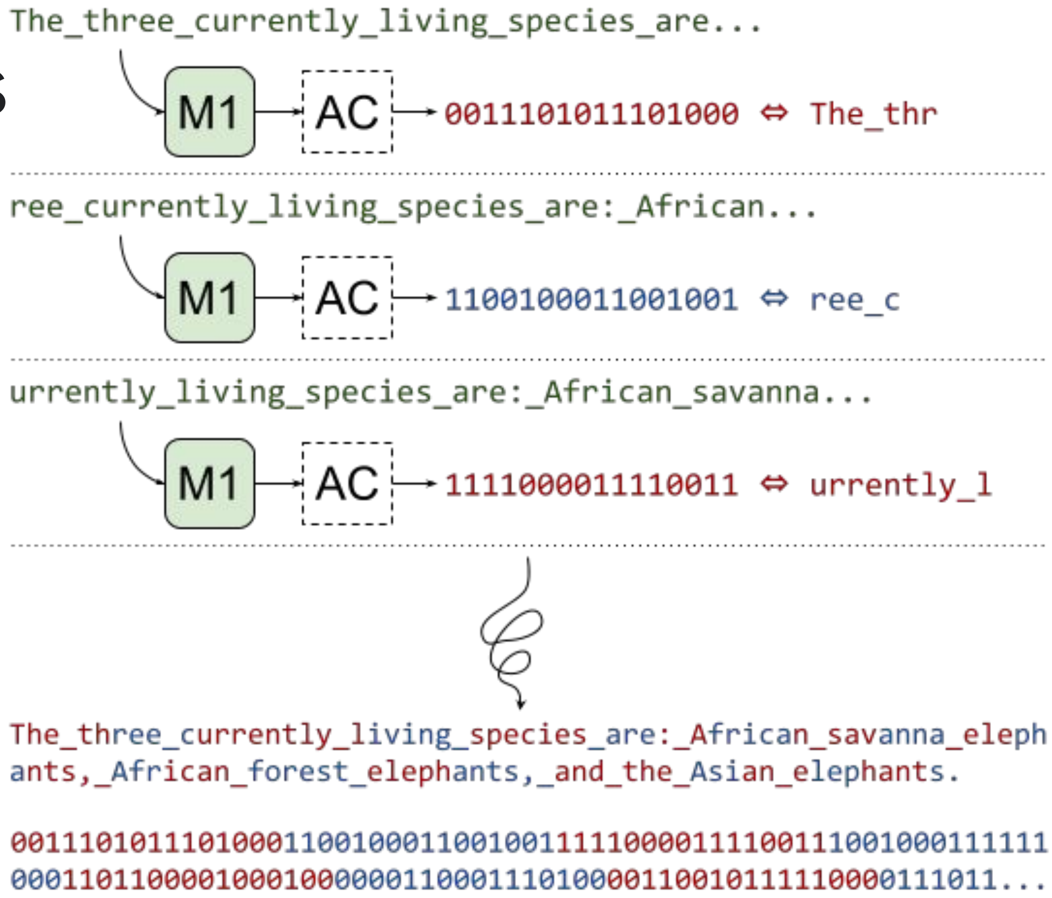
- Compress the whole sequence into a single example.
- Run AC compression based on M1 logits

2b parameter model



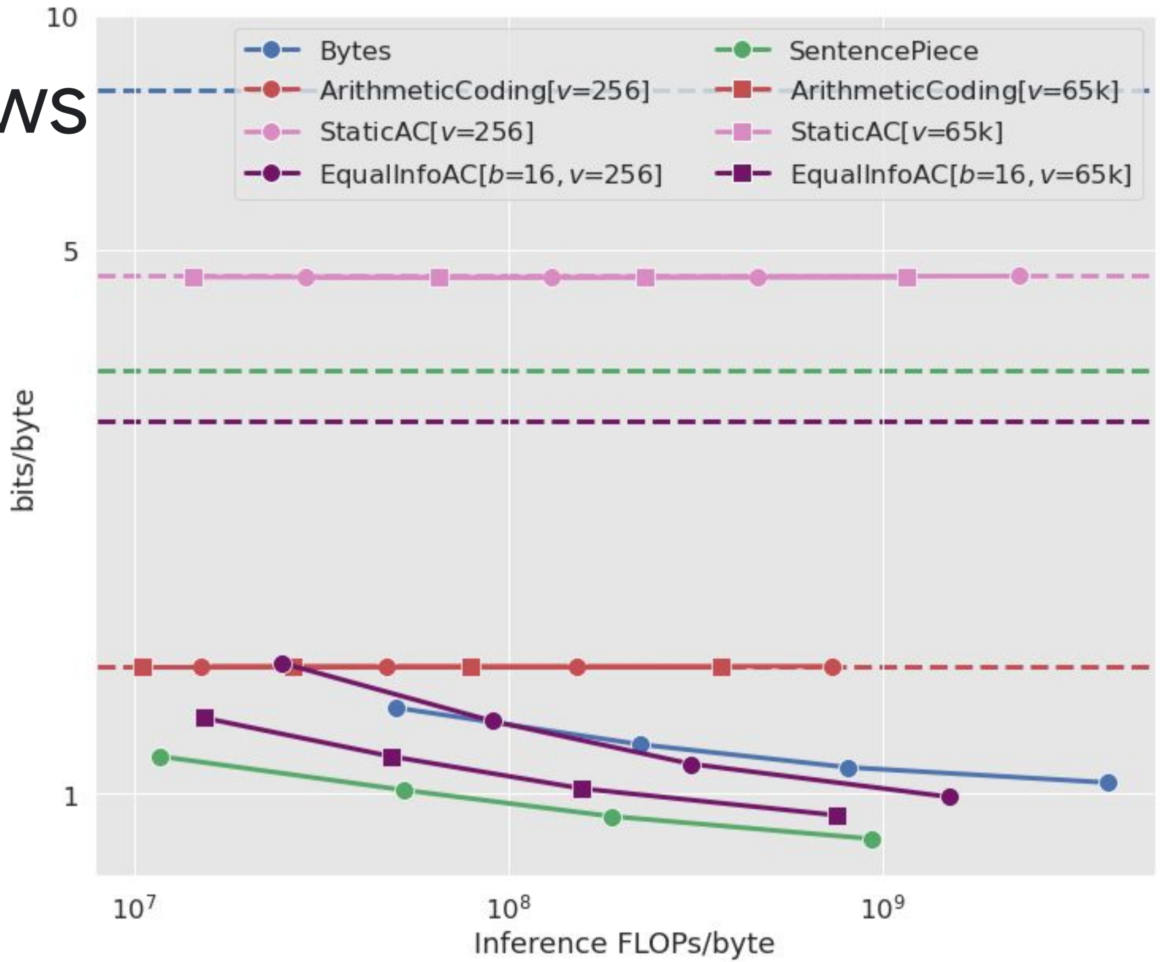
Equal Info Windows

- Can we make it easier for the model to track the AC state over time?
- We reset the AC encoding (and M1's context) when N bits are output
- Entropy based segmentation



Equal Info Windows

- We finally have something that beats the Byte Level baseline
- Approaches the SentencePiece baseline
- Vocabulary is twice as large to boost compression



04

Why is it Hard?

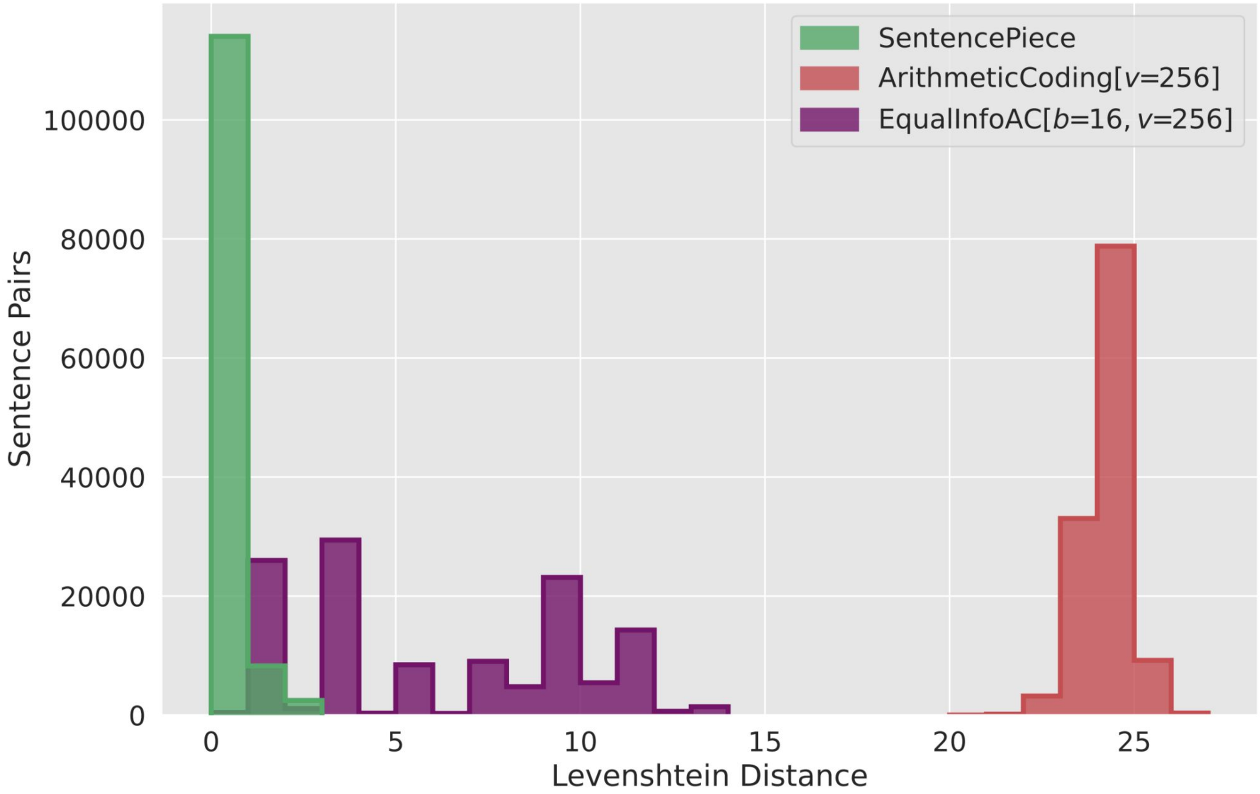
Semantics of Tokenization

- Equal Info Tokens don’t have a clear alignment to words/morphemes
- The same word—“elephants”—is tokenized in multiple ways

Input Text	`The three currently living species are: African savanna elephants, African forest elephants, and the Asian elephants.`
SentencePiece Tokens	`[The] [ three] [ currently] [ living] [ species] [ are] [:] [ African] [ ] [s] [a] [v] [anna] [ elephant] [s] [,] [ African] [forest] [ elephant] [s] [,] [ and] [ the] [ Asian] [ elephant] [s] [.]`
EqualInfoAC [b=16, v=65k] Tokens	`[The th] [ree c] [urrently l] [iving ] [species] [ are] [: A] [frica] [n sav] [anna] [ ele] [pha] [nts, ] [Afr] [ican ] [forest ] [eleph] [ants, ] [and the ] [Asi] [an e] [lep] [hant] [s.]`

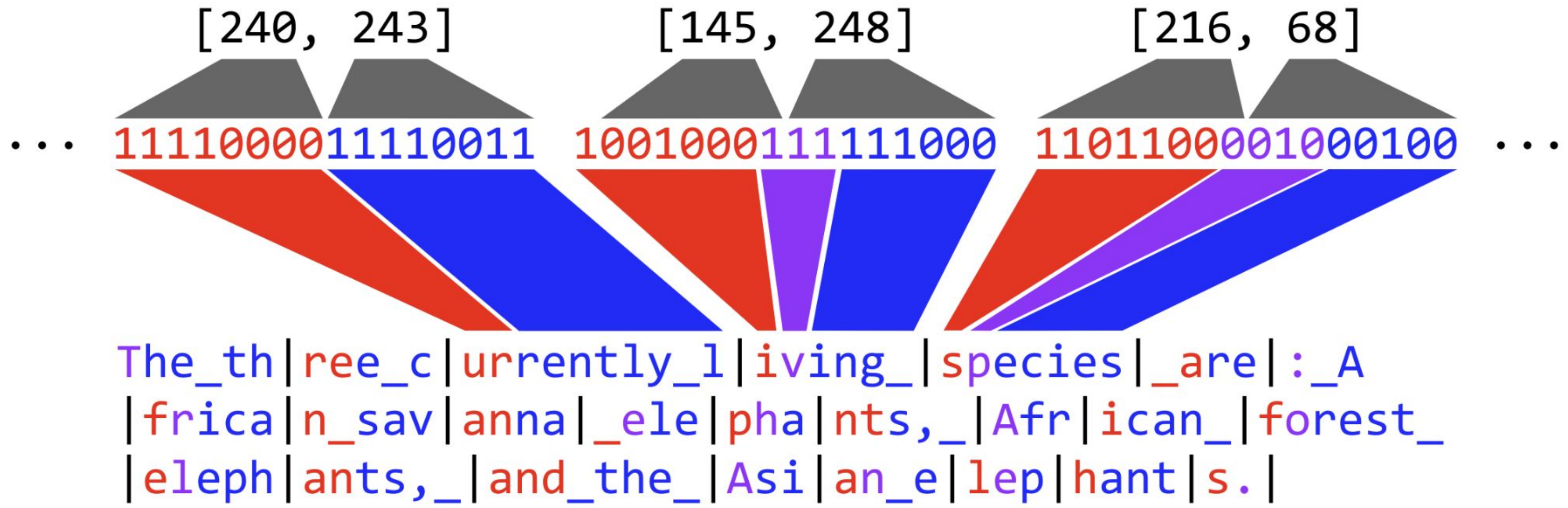
Stability of Tokenization

- Another difficulty is how contextual the tokens are
- A small change to the start of the sentence can cause huge changes in the resulting tokenization



Stability of Tokenization

- Aligning “tokens” to the text is not well defined
- Sometimes bits cross the “token boundary”
- Two pieces of text with a shared prefix can have the same initial token, but the information about the “non equal” text actually lives in both tokens



Takeaways

- Training on AC compressed text doesn't work
- Equal Info Windowing makes it learnable, but it still loses to SentencePiece

Future Directions

- Can we make a new stronger compression algorithm to use as a tokenizer?
  - Without these issues
  - More like SentencePiece
- Entropy Based segmentation

Google DeepMind

Thank You

Brian Lester, Google DeepMind

Published in TMLR; Presented at ICLR 2025