Episode Book

EPISODE #6503

Why Tokenizer Vocabulary Size Is a Hardware Choice

PUBLISHED	8 November 2026 at 14:48 UTC
DURATION	25:22
LISTEN	myweirdprompts.com/episode/tokenizer-vocabulary-size-hardware-choice-50470/
HOSTS	Corn, Herman, Larry, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.26
SOURCES	17 cited by the research (listed at the end)
BOOK GENERATED	9 November 2026 at 08:09 UTC

SYNOPSIS

Before an AI model reads a word, a pre-trained tokenizer chops the text into pieces using byte pair encoding. While vocabulary size is often treated as an arbitrary convention, its optimal dimension varies dramatically depending on whether a model runs on-device or across datacenter GPUs. Looking into merge counts, multilingual additions from Llama 3 to Gemini, and recent findings on Zipf's law, this analysis reveals why vocabulary tuning is fundamentally an infrastructure optimization.

TRANSCRIPT



Corn

Every model you've ever typed a sentence at had that sentence chopped into pieces before the model saw it, by a small program whose mind was made up years earlier. That's the setup Daniel sent us: the tokenizer is itself trained, usually with byte pair encoding or a cousin, on a sample of the corpus. He wants the merge-by-merge mechanics of how BPE builds a vocabulary, how labs pick the size — thirty-two thousand versus one hundred twenty-eight thousand versus two hundred fifty-six thousand — how they decide what languages and code go into its training data, and why the whole thing is basically welded shut the moment pretraining starts.



Herman

And the size question is the one that sounds like a modelling question and turns out to be an accounting question, which is my favourite kind.



Corn

Hold that thought. We should watch the algorithm build a vocabulary first, because the size question only makes sense once you see that vocabulary size *is* the merge count.



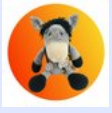
Herman

The framing most people miss is that the tokenizer is a separate model, trained separately, before the language model exists. It defines the alphabet the big model will spend its entire training budget learning to use. Whatever the tokenizer refuses to compress, the model pays for, every token, forever.



Corn

Forever being the operative word.



Herman

One correction to the standard story, though, because it's a good one: byte pair encoding isn't an NLP idea at all. It's a data compression technique from a guy named Gage, nineteen ninety-four — iteratively replace the most frequent pair of bytes in a sequence with a single, unused byte. Sennrich, Haddow and Birch adapted it for word segmentation in twenty-sixteen, and they're admirably blunt about the borrowing. They say it outright: it's



Herman (continued)

a compression technique, we're just merging characters instead of bytes.



Corn

A compression algorithm wearing a linguistics costume.



Herman

And the tension the whole episode lives in is right there in their paper. On one side, strong incentives to minimize the vocabulary of neural models, for time and space. On the other, they want a compact representation of the text itself, because an increase in text length, in their words, "reduces efficiency and increases the distances over which neural models need to pass information." Longer sequences, more steps for information to travel, more compute per document.



Corn

And the original motivation was translation, not chatbots.



Herman

Open-vocabulary translation. Neural translation models back then ran on fixed word vocabularies of thirty to fifty thousand words, but translation is an open-vocabulary problem — German will hand you a compound like Abwasserbehandlungsanlange, sewage water treatment plant, and the fixed vocabulary has no entry for it. Their answer: don't store words, store merge rules that can build any word.



Corn

Sewage treatment plant as an argument for subword units. German does half the work of any NLP paper.



Herman

It really does. Okay, the loop itself. You start with your symbol vocabulary being just the character vocabulary. Each word in your corpus is represented as its characters plus a special end-of-word marker, a little dot, so that after translation you can restore where the word boundaries were — that dot is load-bearing.



Corn

Because if "the" and "theory" both collapse into symbols, you need to know where one stops.



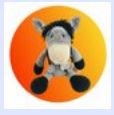
Herman

Then, at every step, you count all the symbol pairs across the corpus and replace each occurrence of the most frequent pair with one new symbol. The pair 't','h' wins early, becomes 'th'. Next round maybe 'th','e' — now you have 'the' as a single symbol. Every merge produces a symbol representing a character n-gram, and frequent n-grams and eventually whole words collapse into single symbols. No shortlist of words, no dictionary of exceptions — the frequency count does all of it.



Corn

Greedy, one merge at a time, most common pair wins.



Herman

Greedy and local. It never revisits. That matters later.



Corn

And here's the arithmetic that makes the rest of the episode make sense — Sennrich et al. state it plainly: the final vocabulary size is the initial character vocabulary plus the number of merge operations, and the merge count, they say, is "the only hyperparameter of the algorithm."



Herman

That's the sentence I'd underline. Vocabulary size isn't a dial sitting next to the merge dial — there's one dial, and the vocabulary is what's left after you stop turning it.



Corn

Which also answers why training is cheap. It never touches the raw corpus.



Herman

Right — because they don't consider pairs that cross word boundaries, the lot runs on a word-frequency dictionary. Each unique word appears once, weighted by how often it turns up. You're counting pairs in a table of maybe a few hundred thousand entries, not a trillion-token corpus. That's why a tokenizer trains in an afternoon on a laptop while the model it feeds takes months on a datacenter.



Corn

The doorman earns less than the act.



Herman

Now, the detail I want to correct before it calcifies — because even Daniel's framing carries it — is the thirty-two thousand figure. Sennrich et al.'s own configurations were fifty-nine thousand five hundred merges for base BPE and eighty-nine thousand five hundred for joint BPE. Not thirty-two thousand. Thirty-two k is a later convention that people read backwards into the origin, like assuming the first car came with seatbelts.



Corn

And it worked — that's the part worth keeping. On WMT fifteen English-to-German and English-to-Russian, the subword models beat a back-off dictionary baseline by up to one point one and one point three BLEU.



Herman

From a compression technique a printer would have recognized. Fine. So if vocabulary size is just merge count, then "how big should the vocabulary be" really means "how many merges do you run, and on what data" — and that's where the labs go in wildly different directions. Llama three is the best documented case. Meta built a one hundred twenty-eight thousand token vocabulary as one hundred thousand tokens lifted from OpenAI's tiktoken tokenizer, plus twenty-eight thousand additional tokens to better support non-English languages.



Corn

Half borrowed from the competition.



Herman

Openly. And the payoff they report: versus Llama two's tokenizer, compression on a sample of English data rose from three point one seven to three point nine four characters per token. Same compute, more text read per token. And they say the twenty-eight k additions improved both compression and downstream performance with, quote, "no impact on English tokenization."



Corn

It means they checked — someone ran the English benchmarks specifically to see if the top-up had cost them anything at home.



Herman

Google sits at the other end. Gemini and Gemma use SentencePiece with a two hundred fifty-six thousand token vocabulary. Nobody official published that, by the way — the confirmed figure comes from someone actually opening the model file for gemini-1.5-pro-002, the SentencePiece model, and reading it: two hundred fifty-six thousand entries, IDs zero through two-five-five-nine-nine-nine, about five hundred six special and control tokens sitting there with a score of zero.



Corn

Verified by taking the lid off.



Herman

Which is very much the spirit of the thing. So — why would anyone go to two hundred fifty-six k? There's a real scaling argument. The Over-Tokenized Transformer work, ICML twenty twenty-five, decouples the input vocabulary from the output vocabulary and scales only the input side up, exploiting multi-gram tokens. Their finding: a log-linear relationship between input vocabulary size and training loss — larger input vocabularies consistently improve the model, regardless of model size, and they match double-sized baselines at no additional cost.



Corn

Bigger alphabet, shorter sentences, model gets more said per step.



Herman

That's the intuition, yes. Now here's the counter-argument, and it's the one that reframes everything. Mittal and colleagues, this August, formalize the lifecycle cost of a vocabulary as training cost plus a weighted inference cost — the training term paid once, the inference term paid per token forever. Their headline result: the inference-optimal vocabulary shifts sixteen times with serving batch size. Thirty-two k at batch one. Five hundred twenty-four k at batch sixty-four and up.



Corn

Sixteen times. From the same model, the same weights — just how you serve it.



Herman

The unembedding matrix read gets amortized across the batch, that's the mechanism. But the punchline is the quality finding: across the optimal range, the spread in bits per byte is under two percent. Their words — quality is approximately invariant, making vocabulary a pure systems optimization with no quality penalty in the measured range. On-device, you want about thirty-two k. Datacenter serving, one hundred thirty to two hundred sixty-two k.



Corn

So the entire thirty-two-versus-two-fifty-six debate is a question about your GPU fleet, wearing a lab coat.



Herman

Corn, quick gut check — does that square with you? You're the one who's suspicious of anything that changes while you're asleep. Does a decision this consequential being, in effect, a procurement choice bother you more or less than if it were a scientific one?



Corn

Less, honestly. A procurement choice can be audited. What bothers me is that nobody framed it as one — Mittal's own line is that vocabulary is "typically fixed at training time based on convention rather than deployment analysis." They're not choosing thirty-two k. They're choosing it because the last model chose it.



Herman

There's also a paper from July, He and colleagues, that says outright all of it is underexplored — selection typically relies on heuristics — and proposes aligning the token distribution with Zipf's law as the criterion. Tested it across natural language, genomics and chemistry, of all things.



Corn

And while everyone argues about the dial, there's a paper from October saying the dial isn't the problem. Shi and colleagues replace the merge machinery itself — their approach doesn't depend on a merge list at all, like BPE, or token probability, like Unigram.



Herman

And the numbers are startling. At two hundred fifty thousand vocabulary on English web text, they beat the o200k base tokenizer's compression by twenty-six percent and qwen250k by thirty. Push to a million entries and they report over sixty percent token-efficiency gain, with vocabulary utilization climbing from fifty-three percent to ninety-seven.



Corn

Half of everyone's vocabulary sitting unused.



Herman

Bigger vocabularies are mostly padding, in other words. Which undercuts the scaling argument from a different direction than Mittal.



Corn

So that's the merge machinery. Anyway, here's Larry with something.

L

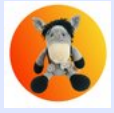
Larry

And now a word from MergeByNumbers, the only tokenizer training kit that lets you merge by numbers. My satisfied customer, let's call him Gerald, no wait, let's call him Bartholomew, told me, Larry, I used to have a vocabulary of thirty two thousand tokens, and my text kept crossing word boundaries like a confused possum, so I fired up MergeByNumbers, and within one afternoon my merge count rose so high my vocabulary practically collapsed into a single symbol. I saved forty percent on time and space efficiency, and my distances over which neural models pass information shrank to the length of a small dog.



Corn

A small dog.



Herman

Corn, I can't. "My vocabulary practically collapsed into a single symbol." That's backwards — each merge adds a symbol. Your vocabulary grows by exactly one per merge. If Bartholomew's merge count rose so high his vocabulary collapsed, Bartholomew has invented negative merging, and the distances he's so proud of don't shrink to the length of a small dog, they shrink because the sequence gets shorter, which is the entire point Sennrich and colleagues were making in twenty-sixteen — it's about the distance information has to travel through the model, not some...



Corn

Herman.



Herman

No, because the forty percent is also unsourced, and...



Corn

It's an ad. Let it go. Bartholomew will be fine.



Herman

The possum line I'll allow.



Corn

Second dial, then. The vocabulary is fixed, but the same afternoon someone also decides what the tokenizer trains on — which languages, how much code. That one you can't undo either.



Herman

The Llama three case is the cleanest documented example, and I want to look at its shape rather than just its existence. Start from tiktoken's hundred thousand — an English-heavy tokenizer, built for English web text. Add twenty-eight thousand tokens for non-English coverage. Check English didn't regress. It's a deliberate, verified patch.



Corn

But do the arithmetic on the patch. Twenty-eight thousand out of one hundred twenty-eight thousand. Roughly twenty-two percent of the vocabulary for every non-English language on Earth, combined.



Herman

Hebrew, Arabic, Hindi, Mandarin, Japanese — all the code, too, if we're counting coverage honestly. It's a top-up, not a rebalancing. The house stays English; other languages get an extension built on the side.



Corn

Which you'd notice if you were the extension.



Herman

The Indian-languages work from last year is the interesting counterexample on method, because it's not about size at all. Das and colleagues compared SentencePiece, BPE and WordPiece across statistical, neural and multilingual translation for agglutinative languages — languages where words are assembled from multiple morphemes, the Abwasserbehandlungsanlange problem as a daily reality. SentencePiece continuously outperformed the others on BLEU for most language pairs, though BPE won some.



Corn

So the algorithm choice itself carries a language bias.



Herman

That's the takeaway I'd defend. The base algorithm, not just the token count, encodes assumptions about how words come apart — and a top-up of twenty-eight thousand tokens can't undo a segmentation strategy that was chosen before those languages were considered. You can give a language more tokens; you can't give it a different theory of its own morphology.



Corn

The token count is the apology. The merge strategy is the insult.



Herman

And there's a domain version of the same trap. Erdogan and colleagues, from January, treat tokenizers as structured compressors and make the point that the corpus mix you fit the tokenizer to determines what it compresses well — and the gains degrade badly under train-test domain mismatch. A tokenizer fitted to web-heavy English will slice through web English and mangle code, math, or low-resource scripts, and it will do so with the same total confidence, because a tokenizer doesn't know what it doesn't compress.



Corn

There was an attempt to get clever across languages, wasn't there — conditioning a target-language tokenizer on source-language tokens from parallel data.



Herman

Vico and Libovický, yes. Mixed result, and honestly the mixed result is more interesting than a win would have been. No improvement in translation quality, but consistent perplexity reductions in language modeling. Their hypothesis for why: the conditional probability estimation scales quadratically with vocabulary size, which creates a data efficiency bottleneck. The vocabulary is too big to condition cheaply.



Corn

The cure costs more than the disease.



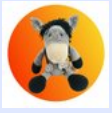
Herman

And that gets us to the freeze, because this is where it stops being a design choice and becomes a prison. The structural reason is boring and absolute: every token ID is a row index into the embedding matrix and a column index into the output layer. The model's entire knowledge is stored in terms of those indices. Change the vocabulary and you change what every index means.



Corn

You're not swapping a part. You're renumbering the building.



Herman

And the scale is what makes renumbering unthinkable rather than merely expensive. Llama three's flagship: four hundred five billion parameters, trained on fifteen point six trillion tokens, at three point eight times ten to the twenty-five FLOPs. Re-indexing that vocabulary means the rows that encode "this token means X in context Y" are now wrong — not outdated, wrong, pointing at tokens that no longer exist. You'd re-learn them from scratch or nothing.



Corn

So it's not that nobody has a clever method. It's that the cost structure has no clever side.



Herman

And the cost asymmetry is the formal version of that. Back to Mittal's equation — training cost paid once, inference cost paid per token forever. The decision is made once, up front, when you know the least about how the model will actually be deployed. That's why convention wins: convention is a hedge against an irreversible decision made under ignorance.



Corn

Now do the compounding. Token count isn't just a bill.



Herman

Right — because the tokenizer sets how many tokens a document becomes, it silently sets compute and context usage too. A tokenizer that produces twenty percent more tokens per document makes the model twenty percent more expensive at inference and retroactively shortens its context window. You didn't change the model's architecture; the same window now holds twenty percent less of the world.



Corn

The tokenizer writes the cheque; the model signs it.



Herman

And here's the part that surprised me, checking whether anyone had solved this: there is no published method for post-hoc tokenizer replacement. Searches turn up nothing on point. The nearest thing anyone's built is the Over-Tokenized Transformer work — which decouples the input vocabulary from the output so you can scale the input side up without touching the output head. That's expansion on one side of the door. Nobody has opened the door.



Corn

The claim "you can't change the tokenizer" rests on cost structure plus an empty shelf.



Herman

Well supported by the first, backed by the absence of the second. I'd phrase it as: nobody has demonstrated a cheap way, and the expensive way is the one you were trying to avoid. Closest thing to a closing thought from the literature is Mielke and colleagues, from twenty twenty-one — "there is and likely will never be a silver bullet singular solution for all applications, and thinking seriously about tokenization remains important." They said that five years ago and it has aged into the understatement of the field.



Corn

We did an episode once on how fine tuning works, and the line I remember defending is that fine tuning changes what a model knows. This is the flip side of it — tokenization fixes what the model can even see. Nobody fine-tunes their way to a new alphabet.



Hilbert

That's the whole of it, Herman.



Corn

Something like that.



Hilbert

Though you've undersold how arbitrary the cut list is.



Corn

The cut list.



Hilbert

The merge list. That's what we called it at the shop, the cut list, because that's what a joiner works off. I've got forty-two of them at home, printed. The one I keep by the desk is from a bloke named Petrov, who ran a typesetting outfit on Jaffa Road. Nineteen ninety-eight he cut his list, and the pair that won merge number eleven thousand and something was a space and a digit. Space-four. Nobody chose it. It was just the most frequent pair that Tuesday. He circled it in biro and I bought the printout off him for thirty shekels when he cleared the place out.



Herman

Wait — a space merged with a digit. That pair shouldn't have been frequent enough to win a round.



Hilbert

It wasn't, if you counted the whole year. But he'd fitted the tokenizer on one week of invoices, and that week every line started with a quantity. So the cut list learned that invoices begin with space-number, and it never learned anything else. He kept it like that, Corn, for years. Cutting it again meant renumbering every job that referenced the list, and the customers' files all pointed at the old numbers. Cheaper to leave the bad merge in forever.



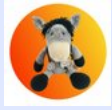
Corn

The invoice tokenizer taxed every document that wasn't an invoice.



Hilbert

Forever. Petrov said the space-four never cost him a job he could name, and then he said that about most of his bad decisions. Thirty shekels. He threw in the biro.



Herman

The thing I keep noticing in that is the training sample. One week of invoices. That's Erdogan's domain mismatch exactly — fit the compressor to one distribution and it mangles the rest, and the mangle is structurally permanent because everything downstream is indexed to it.



Corn

Petrov's defense — that it never cost him a job he could name — is the convention argument in one sentence. Nobody measures the cost of a bad token because the cost never arrives with its own invoice.



Herman

It arrives spread across every job, which is the same as never arriving.



Corn

You know what's stayed with me since the Systems paper — the under two percent quality spread across the whole optimal range. If that holds, then every heated argument about thirty-two k versus two hundred fifty-six k was two teams of engineers arguing about their electric bill and calling it linguistics. The open question I'd leave with is whether the field even notices — Mittal's "convention rather than deployment analysis" suggests the next round of models inherits the last round's answer.



Herman

Shi's result points somewhere more uncomfortable: if a non-merge method beats BPE's compression by a quarter at the same vocabulary size, the frozen-in-amber thing everyone can't change may not be tokenization. It may just be BPE. Which would make the prison a design choice with a real alternative sitting in the October papers.



Corn

The one decision made before the model exists, by a process nobody outside the lab sees, that quietly sets the price of everything the model ever does — and it might have a successor already published.



Herman

Someone should tell Bartholomew. His vocabulary isn't going to collapse itself.



Corn

No. Thanks as always to Hilbert Flumingtop for producing the show. If you want more of this, try episode three, Safetensors or something else; episode seven, Building Custom ASR Tools; and episode twelve, The AI Breakthrough. This has been My Weird Prompts — no question too obscure, no rabbit hole too deep. You can find us at my weird prompts dot com.



Herman

Rate us on Spotify if you feel moved to. We'll be back soon.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **Sennrich, Haddow & Birch, Neural Machine Translation of Rare Words with Subword Units, ACL 2016, pp. 1715 [primary]**

   1725 . Retrieved 2026-11-08.
   https://aclanthology.org/P16-1162.pdf

2. **Llama Team, AI @ Meta, The Llama 3 Herd of Models, dated 2024-07-23 . Retrieved 2026-11-08. [primary]**

   https://arxiv.org/abs/2407.21783v3

3. **OpenAI, gpt-oss-120b model card, created 2025-08-04, last modified 2025-08-26 . Retrieved 2026-11-08. [primary]**

   https://huggingface.co/openai/gpt-oss-120b

4. **rsennrich/subword-nmt reference implementation of BPE . Retrieved 2026-11-08. [primary]**

   https://github.com/rsennrich/subword-nmt

5. **Dan Petrovic, Dissecting Gemini's Tokenizer and Token Scores, 2025-06-05. Retrieved 2026-11-08.**

   https://dejan.ai/blog/gemini-toknizer/

6. **Mittal, Gubrani & Kakollu, Lifecycle-Optimal Tokenization: Vocabulary Size as a Deployment-Regime-Dependent Infrastructure Parameter, 2026-08-11. Retrieved 2026-11-08.**

   https://arxiv.org/abs/2608.11361v1

7. **Huang et al., Over-Tokenized Transformer: Vocabulary is Generally Worth Scaling, ICML 2025, revised 2025-05-23. Retrieved 2026-11-08.**

   https://arxiv.org/abs/2501.16975v2

8. **Shi et al., Counting and Min-Cost Encoding for Tokenization in Large Language Models, 2026-10-01. Retrieved 2026-11-08.**

   https://arxiv.org/abs/2610.01127v1

9. **He, Zeng & Jiang, Pre-trained Models Perform the Best When Token Distributions Follow Zipf's Law, 2025-07-30. Retrieved 2026-11-08.**
https://arxiv.org/abs/2507.22543v1

10. **Cognetta et al., An Analysis of BPE Vocabulary Trimming in Neural Machine Translation, 2024-03-30. Retrieved 2026-11-08.**
https://arxiv.org/abs/2404.00397v1

11. **Mielke et al., Between words and characters: A Brief History of Open-Vocabulary Modeling and Tokenization in NLP, 2021-12-20. Retrieved 2026-11-08.**
https://arxiv.org/abs/2112.10508v1

12. **Das, Choudhury & Mishra, Comparative analysis of subword tokenization approaches for Indian languages, 2025-05-22. Retrieved 2026-11-08.**
https://arxiv.org/abs/2505.16868v1

13. **Vico & Libovický, Conditional Unigram Tokenization with Parallel Data, 2025-07-10. Retrieved 2026-11-08.**
https://arxiv.org/abs/2507.07824v1

14. **Erdogan et al., An Information-Theoretic Perspective on LLM Tokenizers, 2026-01-14. Retrieved 2026-11-08.**
https://arxiv.org/abs/2601.09039v1

15. **Gemma 1 & 2 SentencePiece Tokenizer Reference, 256k vocabulary, Apache-2.0. Retrieved 2026-11-08.**
https://tokenreference.com/tokenizers/gemma2/

16. **OpenModelAtlas, Llama 3 tokenizer reference (128,256 token IDs). Retrieved 2026-11-08.**
https://atlas.klimoza.dev/models/llama-3/tokenize

17. **Gemini token calculator, 256,000-token SentencePiece vocabulary, PaLM 2 lineage. Retrieved 2026-11-08.**
https://www.token-calculators.com/blog/provider-tokens/gemini-google-tokens.html