Episode Book

EPISODE #6538

How Models Score Next Tokens All at Once

PUBLISHED	12 November 2026 at 13:36 UTC
DURATION	25:48
LISTEN	myweirdprompts.com/episode/how-token-probabilities-are-calculated-36597/
HOSTS	Corn, Herman, Clive, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.26
SOURCES	13 cited by the research (listed at the end)
BOOK GENERATED	13 November 2026 at 09:46 UTC

SYNOPSIS

Scoring every word in a vocabulary seems computationally ruinous, but language models and speech recognizers compute every token's probability simultaneously in a single matrix multiplication. Softmax merely normalizes raw scores, while the real cost scales with vocabulary size rather than repeated model evaluations. Examining architectures like NVIDIA's Parakeet reveals how token-and-duration transducers speed up transcription by predicting durations to skip redundant audio frames, paired with greedy argmax decoding.

TRANSCRIPT



Corn

Daniel's latest landed in my inbox next to a cup of tea I never finished, which tells you how long I spent with it. He wants to know how a model actually computes next-token probabilities. Not the polite version. The actual arithmetic. How does a stack of numbers inside the network become "the next word is 'the' with probability ninety-eight percent"? And how is that fast enough to run on a phone?



Herman

And he's not asking idly, because this is a man who spends his days wiring up speech-to-text pipelines. He's poked the thing and noticed that the answer he keeps reading, the one where the model scores every possible next token, sounds ruinously expensive.



Corn

His instinct, as he puts it, is that if predicting one token is one calculation, then weighing all the alternatives should cost n times more, because you'd have to run a separate calculation for each candidate. And then he points at NVIDIA's Parakeet and asks, is that thing really producing a probability for every possible next token at every single step? And is there a way of decoding that just grabs the most likely token and moves on, and is that quicker?



Herman

The lovely thing about the prompt is that the intuition is wrong in a really instructive way. It's wrong about the softmax, and it's accidentally right about something else entirely — beam search. Just not where he thinks it applies.



Corn

The last step of a forward pass is almost embarrassingly small compared to what comes before it. You've got your hidden state — one vector, say a couple of thousand numbers — and at the very end you multiply it by a weight matrix with one row per vocabulary entry. One row per token. Out comes a list of raw scores, one per token in the vocabulary, and then a softmax squashes that list into probabilities that sum to one.



Herman

That's it. Two operations. A linear projection and a normalization. And the key word in that whole description is one. One matrix multiply produces all of the scores at once. Not a loop, not "try each candidate and see." The network never deliberates. It does one wide piece of arithmetic and every candidate's score falls out simultaneously.



Corn

Which is the exact point where the n-times intuition dies.



Herman

Where it dies, yes. The intuition imagines the model standing at a fork, weighing each path by walking it. What actually happens is more like a... a stamp. The hidden state hits the output matrix and the full score sheet is printed in one go.



Corn

You've been waiting all week to use a stamp metaphor.



Herman

I had it written on the whiteboard, actually.



Herman

But here's the part worth being honest about, because it's not free. The attention layers and the feed-forward blocks — the expensive machinery — run once per token no matter how many candidates you're scoring. They don't know or care what the vocabulary is. The output projection is comparatively thin, but "comparatively thin" is doing work in that sentence. For a model with a vocabulary of, say, a hundred thousand tokens and a hidden dimension of a few thousand, that final matrix has hundreds of millions of entries in it.



Corn

So the arithmetic does scale with vocabulary size. Just not with n model runs.



Herman

Right, it's one dense operation whose cost is proportional to vocabulary size times hidden dimension. The distinction that matters is dense versus repeated. A GPU executes a matrix multiply by farming it out across thousands of cores at once. On a phone, it gets vectorized across the CPU's SIMD lanes. The hardware is built for exactly this shape of work.



Corn

Hence a six-hundred-million-parameter speech recogniser running on a laptop.



Herman

Hence that, yes. And the reason people know the output head is a real cost is that there's active research on shrinking it. There's a paper called VQ-Logits that opens with the observation that this final projection often constitutes a substantial portion of the model's parameters and inference compute — and their fix is to replace the full vocabulary-sized matrix with a small codebook. Up to ninety-nine percent parameter reduction in the output layer, six times faster logit computation, at the cost of roughly four percent worse perplexity.



Corn

Six times faster logits. Somebody measured the last two microseconds of the pipeline and found a budget.



Herman

A whole paper's worth of budget! And a second group opened their paper the same way, that large vocabularies make output heads a substantial inference cost in small language models. So nobody's pretending the last layer is nothing. It's a real slice. It's just a slice of one pie, not n pies.



Corn

Okay. Walk me through the softmax itself, because Daniel asked specifically about the probabilities, and I want to be able to picture it.



Herman

The definition is almost tautological when you write it down. It converts a tuple of K real numbers into a probability distribution over K possible outcomes. You exponentiate each score, sum them all, and divide each by the sum. Whatever the raw scores were — negative, huge, tiny — you get K non-negative numbers summing to one. It's the standard last activation when you want a network's output to be a distribution over classes.



Corn

So the softmax knows nothing about tokens.



Herman

Nothing whatsoever. It would happily normalize a shopping list.



Corn

Then the "meaning" of the probabilities is entirely upstream. The softmax is just the accountant.



Herman

The accountant is a good word for it, actually. And here's a detail I like — the transducer paper we'll get to computes these in the log domain, because multiplying long chains of probabilities in the raw domain underflows to zero and the arithmetic dies. Logs turn the products into sums. Boring, essential, exactly like an accountant.



Corn

One thing worth pinning down before Parakeet, because it's the hinge of Daniel's question. When you say the model "computes a probability for every token," someone could hear that as the model seriously weighing every token. Mulling alternatives. Making



Corn (continued)

a considered choice.



Herman

Those are completely different claims, and the gap between them is where greedy decoding lives. But yes — hold that thought, it pays off later.



Corn

The audio prompt Daniel sent a while back about transcription formats — we covered this ground partially there. So: does an ASR model actually do the one-matmul dance, or does it cheat?



Herman

It does it, and Parakeet specifically does it with a twist. First, though, the structure, because this is where Daniel's framing needs a correction that he'd want made. Parakeet is not a language model with a microphone. It's a transducer. Three parts. An encoder chews the audio into higher-level representations. A prediction network — the decoder — carries the text history. And a joint network, the joiner, takes both and generates a probability distribution over the vocabulary. That's from the Token-and-Duration Transducer paper, which is the architecture Parakeet's TDT variants are built on.



Corn

So its "next token" is conditioned on audio frames, not just on text so far.



Herman

And at every single step — every single step — that joiner emits a full distribution over the vocabulary. Including a special blank symbol, which is the transducer's way of saying "nothing to emit here, I'm still listening to this chunk of audio." So the answer to Daniel's yes-or-no is yes, unambiguously. A full distribution over every possible token, every step. That is core to how these models work.



Corn

Now the twist.



Herman

The twist is that the joiner emits two things, not one. One set of outputs for the token, and a second set for the duration — how many audio frames this token or this blank should cover. The paper models them as conditionally independent: the probability of a token and a duration equals the token probability times the duration probability. So alongside the vocabulary softmax, which is big, there's a second, tiny softmax over allowed durations. The paper's example uses a duration set of zero, one, and two frames.



Corn

A three-way bet. The vocabulary distribution has thousands of entries and the duration distribution has three.



Herman

And the duration head is the entire speed story. A conventional transducer processes the encoder's output frame by frame, grinding through every single frame of audio even when the model has known for half a second what it's hearing. During inference, TDT models skip input frames guided by the predicted duration. The model learns to jump. The paper reports up to two point eight two times faster inference than conventional transducers for ASR, two point two seven for speech translation.



Corn

That's the part I find clever, and I resent it. The model isn't faster because the arithmetic got cheaper. It's faster because it learned to stop listening sooner.



Herman

It learned when more listening buys nothing. Which, Corn, is a thing you'd know something about.



Corn

I'll take that as a compliment. But here's my suspicion — Corn's rule of hardware, let's call it: a trick that requires the model to decide how long to attend sounds like it helps the encoder, and does nothing for the part Daniel asked about. The vocabulary softmax still fires every step.



Herman

It does, and you're right that the duration head and the output head are separate savings. But they stack, and the stacking is the point. Skipping frames means fewer joiner evaluations overall, and each evaluation's vocabulary softmax is one parallel operation anyway. Neither one is where the compute lives.



Corn

Here's a number I want on the table, because it makes all of this concrete. Parakeet's speed figure on the open leaderboard is an RTFx of three thousand three hundred eighty. Meaning it transcribes audio thousands of times faster than real time.



Herman

At batch size one hundred twenty-eight, with the caveat that the number moves with audio duration and batch size. But yes — three thousand times faster than the audio is long. And the same model ships with a two-gigabyte minimum RAM requirement and a quantized GGUF build that runs through a native C++ runtime, one line on the command line. The v3 card documents it. A six-hundred-million-parameter model, built with the assumption you'll run it on your own machine.



Corn

The GGUF build — we did an episode on inference formats a while back, the safetensors question, and this is the same debate moved from language models to speech. Quantized local builds versus hosted checkpoints. Herman's answer was "both, and stop asking." Mine was "the one that runs on hardware people own."



Herman

Your answer was "the one that doesn't need an update prompt," which is not the same thing.



Corn

It rhymes.

C

Clive

One-Star Getaways here, and this one's from Kraków. I booked the Vistula Velvet Rest, advertised sea view, and the sea was a painting of the sea, hung slightly crooked, and I'll be honest, the frame was the best part of the stay.

C

Clive

One star. The kettle took eleven minutes and the pillow had a previous guest's opinions in it. If you're after a properly disappointing trip this winter, I recommend it, and tell them Clive sent you, they'll know what it means.



Corn

I'm pricing flights to Kraków in my head right now.



Herman

Don't you dare book the crooked sea. Back to the joiner — Daniel's last question. The approaches that just pick the single most likely token and move on. Those exist, and the answer is sitting in the TDT paper's own pseudocode, which I found delightful. Their Algorithm One, greedy inference for a conventional transducer, loops over frames, joins the encoder and decoder states, and then, verbatim, the line is: take the argmax of the joined output.



Corn

Argmax. The least romantic instruction in all of computing.



Herman

Take the biggest number, take its position, done. If the position is the blank symbol, advance the frame. If it's a real token, append it. The TDT version, Algorithm Two, adds the duration distribution and lets the frame counter jump ahead — the paper calls that the



Herman (continued)

key place that makes TDT inference faster. But the token choice is still an argmax.



Corn

So greedy decoding. It exists, it's not exotic — how much quicker is it really?



Herman

Here's the interesting thing. Greedy is not some niche shortcut. It is the default. The reference configuration people quote. The open ASR leaderboard findings treat greedy as the baseline condition. A thirty-two-million-parameter Turkish recogniser published earlier this month reports its headline numbers under greedy TDT decoding, and benchmarks itself against Whisper large-v3, also greedy. Nobody advertises "we used greedy" because everyone does.



Corn

The industry's open secret is that the fancy part is optional.



Herman

The fancy part is beam search, and the numbers on it are very telling. NVIDIA did a whole optimisation push on transducer beam search — batch operations, tree-structured hypotheses, CUDA graphs, serious engineering. Result: it narrows the speed gap between beam and greedy to only ten to twenty percent for the whole system, while achieving fourteen to thirty percent relative improvement in word error rate.



Corn

Ten to twenty percent. Not ten to twenty times.



Herman

Percent. And only after heroic optimisation — the same paper notes beam search significantly slows transducers because of repeated evaluations of key network components. Unoptimised, it's much worse.



Corn

Here is where Daniel's intuition turns out to be exactly right, just aimed at the wrong target. Beam search is the case where maintaining k hypotheses means running the decoder k times per step. The n-times multiplier is real there. It's just that the decoder — the small network and the vocabulary projection — is the cheap part, so multiplying the cheap part is survivable.



Herman

And the encoder, the expensive part, runs once regardless. That's why the penalty is ten to twenty percent and not ten to twenty times. His intuition was about the vocabulary softmax, where it's wrong, and missed the beam search, where it's correct. The correction is almost surgical.



Corn

I want to check something against your clinician's brain, Herman. You spent decades explaining small probabilities to frightened people. Beam search spends ten to twenty percent more compute to improve word error rate by fourteen to thirty percent, relative. Relative means the absolute number is smaller than it sounds. When you translated risk for patients, how did you decide when a relative improvement was worth acting on?



Herman

By asking what the absolute risk was, always. If the error rate's already low, a thirty percent relative improvement is a rounding error on a rounding error. If it's high, or if the domain is high-stakes — medical dictation, courtroom transcripts, anything where one wrong word costs real money or real harm — then you buy it. My honest clinical instinct here: a machine transcribing routine notes can carry a modest error rate because a human reviews it. A machine doing anything unsupervised, you start counting words differently.



Corn

That's the pressure I was feeling toward. The trade isn't decided by the arithmetic. It's decided by whether anyone reads the transcript afterward.



Herman

And there's a middle path the production world is converging on, which I find quietly convincing. NGPU-LM does n-gram context biasing — nudging greedy decoding with a small language model's preferences — with under seven percent overhead, and it closes



Herman (continued)

more than half the accuracy gap between greedy and beam search for out-of-domain audio. So the practical answer isn't "beam search." It's "greedy plus a cheap correction."



Corn

Greedy with a tutor looking over its shoulder.



Herman

Cheaper than hiring the whole examination board, which is what beam search is.



Corn

Back to the distinction I flagged earlier, because I promised it pays off. "It computes a probability for every token" versus "it seriously considers every token." I've been assuming greedy's argmax is choosing from a genuine contest.



Herman

It usually isn't, and there's research showing exactly that. Wullach and Chazan looked at fine-tuned wav2vec 2.0 and HuBERT models and found the top-layer logits yield highly confident predictions — meaning the distribution is extremely peaked, one token towering over everything. And when they relaxed that overconfidence, by aggregating information from several layers instead of trusting only the last one, they got word error rate reductions up to ten percent and character error rate reductions up to twenty-two percent under beam search.



Corn

So the model isn't weighing alternatives. It's announcing one and filing paperwork on the rest.



Herman

The paperwork is real — the probability for every token exists as a number. But a probability of ninety-nine point whatever and then a long tail of effectively zero is not a deliberation. That's also why beam search helps less than you'd hope: if the top candidate is already at ninety-nine percent, exploring the other beams is mostly sightseeing.



Corn

The full distribution Daniel's imagining is mostly a formality.



Herman

On well-trained models in-domain, mostly yes. Which loops back to why greedy is the default. The distribution is peaked, the argmax is nearly always right anyway, and the rare cases where it isn't cost less to patch afterward than to catch live.



Corn

And that connects to the leaderboard finding — that the CTC and TDT decoders, the fast greedy-friendly ones, offer superior throughput, making them better suited for long-form and batched processing, while the heavier attention-based decoders win on raw accuracy. Two products for two markets.



Herman

The batch transcription market — somebody's dumping ten thousand hours of audio and wants it by Tuesday — and the single-pass dictation market, where someone's talking to their laptop and wants words on screen with no lag. Herman Poppleberry, reporting for the throughput camp.



Corn

The lag camp. Nobody dictating has ever thanked a beam.



Herman

One more detail from the model cards that closes the loop on the whole question. Parakeet's hosted API accepts a max-alternatives setting, and the default is one. One transcript. Not a ranked list, not a distribution — a single answer. The distribution exists inside the machine, and the door out of the machine is one word wide.



Hilbert

Herman. The argmax line in that transducer paper is not the whole greedy step. There's a blank check before it. The loop reads the frame, checks the blank, and only then takes the argmax. Small thing, but you said it verbatim and it wasn't verbatim.



Herman

He's right, too. The blank test comes first.



Corn

There's a story behind this correction. There usually is.



Hilbert

Late eighties. I was doing alignment work at a relay station outside Nairobi, and the man on the next bench, Ferris, built a shortwave decoder in a biscuit tin. He claimed it weighed every possible character before printing. It did not. It snapped to the strongest signal and moved on. I told him it was cheating. He told me that was exactly what the professionals did, and that was the end of the diplomacy.



Corn

You kept a log.



Hilbert

I kept a log of every transmission his decoder mis-snapped, indexed by weather. Humidity moves the shortwave bands, and Ferris's tin never accounted for it. Two hundred and eleven entries. The odd one was the listening. His decoder got fast not by skipping the



Hilbert (continued)

alternatives — Herman had that right — but by deciding how long to listen before it committed. Variable dwell, he called it. I notice your Parakeet does the same thing with its duration head, and I have been waiting this whole episode for one of you to say so.



Herman

He arrived at it independently. From a biscuit tin and an argument.



Hilbert

The claws made the log awkward. You try writing two hundred and eleven entries in pencil with these.



Corn

Is Ferris still with us, as it were?



Hilbert

Builds decoders in Keswick. Sends a card at Christmas.



Herman

Variable dwell, decided per decision. That's the TDT insight stated from the other end — the paper gets speed by learning durations; Ferris got speed by hardcoding an impatient dwell. Same bottleneck, opposite provenance.



Corn

And the weather index is the part I like — a duration head that doesn't know it's a duration head.



Corn

Cutting-room floor, before we wrap. The thing I couldn't fit anywhere: nobody publishes Parakeet's actual vocabulary size. The model cards don't state it, the paper doesn't state it, and the duration set is only given as an example in a figure. For a machine whose entire job is emitting a distribution over a vocabulary, the vocabulary is unlisted.



Herman

And the RTFx figure is end-to-end. No published breakdown of how much time goes to the encoder versus the joiner versus that final softmax. We know the output head is a substantial cost in language models from the papers that shrink it — but nobody's measured it specifically for ASR. The extrapolation is plausible, not established.



Corn

So here's the thing I'm left holding, and it's mostly from Hilbert's biscuit tin. The research we have on greedy versus beam — that ten to twenty percent figure — was measured on GPUs with CUDA graphs. Nobody has measured it on a phone. And a phone is exactly where Daniel's question lives. That's the study I'd read the moment it exists.



Herman

Produced, as ever, by Hilbert Flumingtop. The log stays in the drawer, Hilbert, that was the arrangement.



Corn

If you want more of this, try episode three, Safetensors or something else; and episode four, If Your Voice Ages, Does Your Fine-Tune Become Useless. This has been My Weird Prompts. If the show earns a spot on your commute, a review on Spotify tells the algorithm to send more of you our way. Find everything at my weird prompts dot com. We'll be back soon.



Herman

See you tomorrow, everyone.

SOURCES

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

1. **NVIDIA, Hugging Face model card, created 2025-04-15, last modified 2026-06-29, retrieved 2026-11-12. [primary]**
   https://huggingface.co/nvidia/parakeet-tdt-0.6b-v2

2. **NVIDIA, Hugging Face model card, created 2025-08-04, last modified 2026-08-05, retrieved 2026-11-12. [primary]**
   https://huggingface.co/nvidia/parakeet-tdt-0.6b-v3

3. **Xu et al., Efficient Sequence Transduction by Jointly Predicting Tokens and Durations (TDT), v2 revised 2023-05-29, retrieved 2026-11-12. [primary]**
   https://arxiv.org/abs/2304.06795

4. **Wikipedia, last edited 2026-11-10, retrieved 2026-11-12.**
   https://en.wikipedia.org/wiki/Softmax_function

5. **Grigoryan et al., Pushing the Limits of Beam Search Decoding for Transducer-based ASR models, Interspeech 2025, submitted 2025-05-30, retrieved 2026-11-12.**
   https://arxiv.org/abs/2506.00185v1

6. **Bataev et al., NGPU-LM: GPU-Accelerated N-Gram Language Model for Context-Biasing in Greedy ASR Decoding, Interspeech 2025, submitted 2025-05-28, retrieved 2026-11-12.**
   https://arxiv.org/abs/2505.22857v1

7. **Grigoryan et al., FlexCTC: GPU-powered CTC Beam Decoding With Advanced Contextual Abilities, ASRU 2025, revised 2025-08-13, retrieved 2026-11-12.**
   https://arxiv.org/abs/2508.07315v2

8. **Shao et al., VQ-Logits: Compressing the Output Bottleneck of Large Language Models via Vector Quantized Logits, revised 2026-09-18, retrieved 2026-11-12.**
   https://arxiv.org/abs/2505.10202v2

9. **Kadav et al., Softmax Reparameterization for Output-Head Quantization, ICLR 2027 submission, revised 2026-09-28, retrieved 2026-11-12.**
   https://arxiv.org/abs/2609.31291v2

10. **Wullach & Chazan, Enhancing Speech Recognition Decoding via Layer Aggregation, revised 2022-04-05, retrieved 2026-11-12.**
    https://arxiv.org/abs/2203.11325v2

11. **Srivastav et al., Open ASR Leaderboard, revised 2026-03-30, retrieved 2026-11-12.**
    https://arxiv.org/abs/2510.06961v4

12. **Teke, ufakzeka-kulak: Building and Evaluating a 31.6M-Parameter Turkish Speech Recognizer from Scratch, submitted 2026-11-02, retrieved 2026-11-12.**
    https://arxiv.org/abs/2611.03605v1

13. **Futami et al., Non-autoregressive Error Correction for CTC-based ASR with Phone-conditioned Masked LM, Interspeech 2022, retrieved 2026-11-12.**
    https://arxiv.org/abs/2209.04062v1