Episode Book

EPISODE #6497

Why Speech Recognition Confidence Scores Lie

PUBLISHED	8 November 2026 at 14:35 UTC
DURATION	26:36
LISTEN	myweirdprompts.com/episode/speech-recognition-confidence-scores-44329/
HOSTS	Herman, Corn, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.26
SOURCES	18 cited by the research (listed at the end)
BOOK GENERATED	9 November 2026 at 08:09 UTC

SYNOPSIS

Speech recognition models like Whisper and Parakeet compute internal probabilities for every token, but turning those numbers into trustworthy word-level confidence is surprisingly brittle. Softmax saturation, pooling choices across sub-words, decoding fallbacks, and blank frames in CTC architectures distort confidence scores across the board. The mechanics behind word confidence calculations, why fine-tuning often worsens calibration, and how recent beam search inspection techniques attempt to quantify a model's true uncertainty are explored.

TRANSCRIPT



Herman

A transcript with a straight face. That's what every speech recognition model hands you — the same flat text whether it heard you perfectly or it took a guess in the dark.



Corn

Daniel's prompt this week is about what's missing from that flat text. Every mainstream ASR stack produces some internal probability signal — token probabilities out of Whisper's decoder, frame posteriors from CTC models like Parakeet, word scores in NeMo — but the transcript you get back throws all of that away. He wants to know how word-level confidence is actually computed underneath, whether ninety percent confidence means right nine times out of ten, why fine-tuned models seem to get more overconfident rather than less, and what you could build if you had a score you trusted — flagging words for human review, deciding when to fall back to a bigger model.



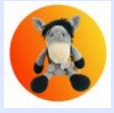
Herman

And the twist buried in there is that the numbers already exist inside the model. Nothing new has to be invented. The question is whether they mean anything once you pull them out.



Corn

Two models, same transcript, same shrug. One of them was guessing the whole time and the page looks identical.



Herman

That's the core tension. Whisper's decoder, Parakeet's CTC head, the transducer models in NeMo — they all emit probabilities for every token or frame they produce. The pipeline that turns those into clean text just... drops them. And the literature here is unusually blunt about what happens if you keep them. Raw softmax probabilities as confidence: one technical blog from NVIDIA sums it up as "fast, simple, and likely useless."



Corn

That's a review you frame.



Herman

It's a great line, because it's true. But the nuance is what makes this worth an episode. The failure isn't uniform. Whisper is actually decently calibrated on clean speech — and then it collapses under noise, which is precisely when you need the confidence signal most. Calibration here is conditional on conditions.



Corn

So the number lies to you exactly when you'd consult it.



Herman

In the worst cases, yes. We'll get to the mechanism first, then calibration, then why fine-tuning makes it worse, then what a score you could actually trust would buy you.



Corn

Start with Whisper, since that's what most people are touching.



Herman

Whisper is an encoder-decoder transformer. The decoder predicts one sub-word token at a time, and at each step it produces a full probability distribution over the vocabulary — softmax over logits. That's the raw signal. And the word-level confidence, if you go pull it from OpenAI's own timing module, is exactly one line of numpy: the arithmetic mean of the token probabilities that span the word.



Corn

A mean.



Herman

A plain mean. So take a two-token word. One token the model was nearly certain about — probability 0.99. The other token was a coin flip — 0.5. The word's confidence is about 0.75. Looked perfectly respectable on the page.



Corn

That's the word you wanted flagged, and it scored a B-minus.



Herman

Mean pooling lets a confident sub-token mask an uncertain one. And here's where it gets properly messy — that's not the only pooling rule in circulation. The trained confidence module baseline in one of the Whisper papers takes the score of the *last* token of each word. An earlier evaluation paper found *sum* pooling worked best. Three different rules for the same quantity, and nobody involved claims the choice is neutral.



Corn

Three laboratories, one number, no agreement.



Herman

And the choice isn't academic — it changes which words get flagged. Mean pooling smooths, sum pooling is dominated by the worst token, last-token pooling bets that the end of a word is where the model has committed. Those will disagree on real transcripts.



Corn

Can you make that concrete? Walk me through a word where they actually split.



Herman

Sure. Take a three-token word where the token probabilities are 0.95, 0.9, and 0.4. The mean is about 0.75 — respectable, passes most thresholds. The product — which is what sum pooling on log probabilities amounts to — is about 0.34, which trips everything. And last-token pooling reads 0.4, the worst of the three readings. Same word, three flags or no



Herman (continued)

flags depending on which line of code you inherited.



Corn

So two teams running "confidence flagging" on the same audio can flag entirely different words, and both think the other one's pipeline is broken.



Herman

And neither is wrong, exactly. Neither is right either. That's the state of it.



Corn

Before we leave Whisper — people talk about avg_logprob and compression_ratio and no_speech_prob like they're confidence scores.



Herman

They're not, and this is worth being firm about, because people build on them anyway. A faster-whisper maintainer put it about as plainly as you can: "They are decoding diagnostics, not four independent confidence probabilities." Compression ratio is literally the gzip ratio of the decoded text — it catches the model going into a repetitive hallucination loop, because "thank you thank you thank you" compresses beautifully. No speech probability is the decoder's vote that there's nobody talking. Average logprob is the mean of the log probabilities. Useful signals, all three, none of them calibrated, all of them sensitive to language, length, and decoding settings.



Corn

Sensitive how? Give me one of those sensitivities.



Herman

Length is the easy one. Average logprob is a mean over tokens, so a long segment dilutes one catastrophic token into a number that looks fine. A segment with a single coin-flip token in a sea of certainty can score the same as a segment that was shaky throughout.



Herman (continued)

The aggregation hides the thing you were aggregating toward.



Corn

So it's a smoke detector, not a thermometer.



Herman

Nice distinction, actually. The defaults tell you how they're used: compression ratio threshold at 2.4, logprob threshold at minus one, no speech threshold at 0.6. If a segment trips one of those, Whisper sets a fallback flag and re-decodes at a higher temperature. There's a whole ladder — zero, then 0.2, 0.4, 0.6, 0.8, 1.0.



Corn

Six rungs of "try being less sure."



Herman

It's a crude confidence-gated retry, already shipped, that almost nobody realizes they're running. And the interesting thing about raising the temperature is what it does to the probabilities you might have wanted to keep — higher temperature flattens the distribution, so the *retry* is guaranteed to report less confidence than the attempt that tripped the alarm, even if it lands on the right answer. The ladder can't tell you it succeeded.



Corn

So the retry fixes the transcript but poisons the very signal we're here to discuss.



Herman

If you're capturing probabilities during a fallback decode, you're capturing numbers from a model that was deliberately made less certain. Nobody warns you about that in the docs.



Corn

Now — CTC models, which is the Parakeet family. Completely different starting point.



Herman

Right. The model emits a probability distribution over the vocabulary for every frame of audio, frames at maybe a hundred per second. So confidence starts at the frame level: for each frame you take the max posterior over the vocabulary, or an entropy-based measure — how spread out the distribution is.



Corn

Frame-level first, then you have to climb up to words.



Herman

Right, and NeMo, which is where Parakeet's confidence machinery actually lives, makes every step of that climb a config option. Max probability or entropy — and if entropy, which entropy: Gibbs, Tsallis, or Rényi, with an entropic index that behaves like temperature, and a normalization mapping into the zero-to-one range. Then the aggregation from frames to tokens to words: mean, min, max, or product. You can even exclude blank frames from the pooling.



Corn

Four aggregation choices, three entropy flavors. That's a menu, not a method.



Herman

It's honest, in a way — the library is telling you nobody knows the right answer, so here are knobs. The pull request that added all this to NeMo's Hypothesis object — frame confidence, token confidence, word confidence — is from a few years back now. And one thing worth saying plainly, because people get it wrong: Parakeet's model card advertises word-level *timestamps*. It says nothing about confidence scores. If you want confidence out of Parakeet, you're configuring it in the NeMo decoder, not reading the card.



Corn

That's a misconception people will actually act on.



Herman

They do. And the blank-frame question is sneakier than it sounds, by the way. In CTC, the blank token is the model's way of saying "no output this frame" — and it's often the *most* confident prediction on the tape. If you pool blanks in with everything else, a stretch of silence between words can carry gorgeous, confident numbers that have nothing to do with any word. That's why the exclude option exists.



Corn

So your confidence score can be dominated by the parts where the model said nothing.



Herman

If you configure it carelessly, yes. Then transducer models — RNN-T — have their own headaches, mainly that sub-word tokenization isn't unique. The same word can be split into tokens different ways across runs, and tokens can get deleted during decoding. The Google work on this, Wang and colleagues at ASRU, handles it with a lookup-table trick mapping sub-word posteriors to word scores, plus a small neural confidence module that exploits the time information attached to each recognized word to keep the compute down.



Corn

Because a word in a transducer carries its own timestamp baggage.



Herman

It does, and that turns out to be a feature — you know which frames to look at. Then the newest angle, from this summer at ICASSP: instead of hand-engineering features and tying yourself to one architecture, you mine the beam search itself — the scores and ranks of tokens *within* a hypothesis. The runner-up candidates are the evidence. If the second-best token was nearly as good as the first, the model was guessing.



Corn

The beam search is the model's own doubt, written down.



Herman

That's a better one-line summary than the paper manages. And it's architecture-independent — hybrid, transducer, whatever — which is why the results are strong. Word-level calibration error around a third of one percent, against about one and two-thirds for raw softmax. But the real gain is on the *worst* bin — maximum calibration error drops from roughly eighteen percent to eight. The worst case, which is the case that matters when a threshold is tripping.



Corn

Say the temperature-ladder thing again. Zero, then 0.2...



Herman

0.4, 0.6, 0.8, 1.0. Six temperatures.



Corn

The model's last confession is a coin.



Herman

So that's how the number gets made. Now the question Daniel actually asked: does it mean anything? Does ninety percent mean right nine times out of ten?



Corn

My guess is no, or this is a short episode.



Herman

Your guess is correct, and the reason is structural. Every loss function in mainstream end-to-end ASR — cross-entropy, CTC, RNN-T, the whole maximum-likelihood family — reaches its minimum when the target token's probability is pushed to one and everything else is pushed to zero. The training loss *rewards saturation*. So the model learns to speak in certainties whether it has grounds or not. Even on a wrong prediction, the winning token often carries probability above 0.9.



Corn

The loss grades on conviction, not accuracy.



Herman

And it's architecture-independent — a simple recurrent model can be just as overconfident as a conformer. It comes from the loss, not the wiring. Which also answers why nobody fixed it by accident: making models bigger and better doesn't touch the incentive.



Corn

So the raw probability isn't a measure of confidence at all. It's a measure of how saturated the softmax got.



Herman

That's a better sentence than mine. Yes. And there's a second property people run together with calibration: discrimination. Calibration is whether the *number* is honest — of all the words scored 0.9, are about ninety percent right? Discrimination is whether the score *separates* right from wrong at all — could you draw a threshold anywhere useful? You can have a perfectly calibrated score that's useless for flagging, because it never separates.



Corn

A thermometer that's accurate but only measures the average of the room.



Herman

Keep going with that, it's good. Whisper's raw probabilities fail both, but the discrimination number is the more damning one. Even on clean audio, high signal-to-noise, Whisper's equal error rate starts around twenty-five percent. A quarter of suspect words mis-sorted, in one direction or the other, on your *best* audio.



Corn

One in four, on a good day, with a clean microphone.



Herman

And that's before the noise shows up. There's a paper from ASRU last year that measures this properly — noise injected at graded levels, calibration metrics at every band. And the pattern is interesting: Whisper holds decent calibration on clean and moderately noisy input. Even when accuracy degrades, it retains what the authors call a degree of self-awareness — it sort of knows when it's likely to be wrong. That holds down to roughly minus five to minus ten decibels.



Corn

And below that?



Herman

It collapses. At the worst band, minus eighteen to minus fifteen, the calibration error is around 0.24 and the negative calibration entropy goes *negative* — worse than an uninformative baseline. But here's the detail I keep coming back to: word error rate starts climbing below zero decibels. Calibration doesn't break until minus five to minus ten.



Corn

So there's a gap. Between zero and minus five, accuracy is already sliding and the confidence still looks trustworthy.



Herman

The dangerous zone. If you're gating human review on confidence, that band is where the errors slip through dressed as certainties. And at low SNR overall, the overconfident mass — incorrect tokens scored above 0.7 — runs ten to twenty percent. At the worst band, over a fifth of wrong tokens are wearing a confident number.



Corn

A fifth of the lies come in a nice suit.



Herman

And to make the failure vivid: think of a busy hospital corridor, a hallway conversation recorded on a phone. That's not minus eighteen decibels — that's ordinary. The band where confidence is already quietly lying isn't an extreme condition, it's Tuesday.



Corn

Now, Daniel's fine-tuning question, because it follows directly. If raw scores are already overconfident, fine-tuning pours fuel on it — and I want to flag that this part is mechanism, not a measured before-and-after. Nobody has published a clean experiment that fine-tunes a model and measures the calibration damage.



Corn

You're sure about that?



Herman

Reasonably. I haven't seen one, which I found surprising. But the mechanism is solid. Same loss, same saturation incentive — and now you're taking many more gradient steps on a small, narrow, clean dataset where the target is nearly always achievable. The model learns "on this data I am right." Softmax saturation deepens, and it carries that saturation out of distribution.



Corn

It's a small pond, and the model memorizes that it's a big fish.



Herman

And everything outside the pond gets the same confident voice. Which is the exact framing of the C-Whisper paper — this is why trained confidence modules exist at all, because using the output probabilities directly "often suffers from overconfidence," high probabilities on words that are actually wrong.



Corn

You take a general model, you fine-tune it on your medical audio, your legal audio — and you've just made the confidence numbers worse on everything you didn't fine-tune on.



Herman

Which is most of the tape, in practice. Nobody fine-tunes on the noisy stuff — you fine-tune on the clean curated corpus because that's what you have. So the distribution the model is *least* prepared for is the one it now speaks about with the most conviction.



Corn

Now — the obvious fix. Temperature scaling. Divide the logits by something, unsaturate the softmax.



Herman

The obvious fix, and the literature is remarkably united that it doesn't work. Global temperature scaling, one temperature for everything: NVIDIA's own blog says it "does not significantly improve confidence." Wang et al. tried T equals two and got, their words, a small improvement. One temperature can't fix a localized problem.



Corn

One thermostat for a house where only one room is on fire.



Herman

The fix that works is selective. The ASRU paper trains a tiny classifier — three point four million parameters, small enough to be an afterthought — whose only job is to decide *which* tokens are overconfident. Then only those get rescaled, with the temperature constrained to be greater than one, so scaling can only ever reduce confidence, never inflate it. The classifier flags ten to fifteen percent of tokens per batch, recall around 0.7, precision around 0.3 — so it's catching most of the liars and a fair number of honest ones too.



Corn

Thirty percent precision. It's a dragnet, not a detective.



Herman

A cheap dragnet. And the payoff is a fifty-eight percent relative reduction in calibration error in the noisy band. Overconfident mass drops from eleven percent to under seven.



Corn

Not bad for a model the size of a rounding error.



Herman

Cheap, selective, one-directional — those are the three properties that make it work, and every one of them is what global temperature scaling lacks. And notice what the classifier gets to look at, because it matters: it's not just the probability. It sees the features around the token — and the model it's attached to stays frozen. You're not retraining the ASR system, you're bolting on an auditor.



Corn

Which is why it's deployable at all. Nobody wants to re-run a benchmark suite because they touched the confidence score.



Hilbert

Medieval Pottage Facts. Salt for the pot had to be bought, and in years of dearth even salt grew dear. The pot was bland and the year was bad.



Herman

The audit office on Hanevi'im. My brother-in-law Yankel spent half of last winter in there with a forms query, and he holds an opinion on confidence scores that I have been waiting the whole episode to relay, because he is not to be trusted, but he is not wrong either. Yankel says every computerized confidence number he has ever been shown was a confidence that had already been decided. The machine prints the number after the machine has chosen, he says, so it is not a confidence, it is a receipt. Herman will tell you the receipt can be recalibrated. My opinion of Yankel's opinions is on record. But on the receipt, I think the old man has it. He got a letter from the records office in the end, and the first thing it said was that their previous letter had been issued in error and should be disregarded. No number on either of them.



Corn

He waited the whole episode for "it's a receipt."



Herman

His grocery receipts also say thank you, but that's not a confidence either.



Corn

The error-confirmation letter is a nice touch, though. "Please disregard the previous confidence."



Herman

The gold standard use case is medical transcription. Wang and colleagues at Google motivated their whole paper with it — "where errors can impact patient care and the confidence estimate could be used to alert medical professionals to verify potential errors in recognition." That's the picture: the transcript goes in flagged, not clean.



Corn

The doctor can't check the whole transcript. The score tells her which twenty words to read.



Herman

And in the clinic I'd take an imperfect score over a clean page every time, because the clean page tells you the system is fine and the score at least admits doubt. NeMo's docs say it in one line — confidence "helps applications decide when to trust ASR output and when to request human review."



Corn

And the economics of that actually pencil out, right? If the reviewer reads only the flagged two percent instead of the whole document, you can afford to have a much more careful human in the loop.



Herman

That's the quiet argument underneath all of it. Human review doesn't scale to full transcripts; it scales to *pointed-at* transcripts. The score is what does the pointing.



Corn

Then the fallback case. Route the hard audio to a bigger model.



Herman

"Selective re-recognition," the papers call it, and here's the honest gap: nobody has published a measured version of it. Route the bottom ten percent of words to the big model, measure the word error rate gain — the experiment doesn't exist. What exists is Whisper's temperature ladder, which is that idea at segment level with no routing at all. The design is sitting there waiting.



Corn

Nobody's built the two-tier thing.



Herman

The parts are on the shelf and the assembly hasn't been done. Beyond review and fallback: data selection. Confidence as a criterion for what to annotate next — active learning, where you label the words the model was least sure about and retrain. There's a line of work showing a better confidence criterion for selection gives a real reduction in word error rate, which makes sense, because if you're choosing what to label, choosing badly is the whole cost.



Corn

And that connects back to the calibration problem in an uncomfortable way — if the confidence scores are overconfident, your active learning is systematically labeling the wrong things.



Herman

It's overconfidence all the way down. The same broken number that misleads the reviewer misleads the annotation budget. Fixing calibration isn't just about the display layer; it changes what the next generation of models gets trained on. And uncertainty propagation — if the transcript feeds a speech translator, the doubt should travel with it. A translator that sees a coin-flip word can hedge, or ask, or flag downstream. A translator that sees a confident word commits.



Corn

And the hallucination case.



Herman

Handled with heuristics, mostly. Compressibility thresholds, no speech probability, low average logprob — the community proposal is that a compression ratio below two is real speech and anything above two-ish is probably a hallucination, and there's a standing request for a hard threshold that discards rather than retries. With a caveat worth keeping: no speech probability on its own is unreliable. People have watched it go to 1.0 on perfectly fine speech.



Corn

The smoke detector that goes off during dinner.



Herman

My guess is the cascade, because the parts exist and the experiment doesn't. First clean before-and-after on fine-tuning damage, first measured confidence cascade — whoever does those, that's the paper everyone cites for the next five years. Until then, every confidence number you're shown was decided after the fact. Check what was in the room when it was measured.



Corn

On that cheerful note, producer credit to Hilbert Flumingtop, who keeps this show running and Yankel's opinions on file. If you enjoyed this one, try episode nine, Benchmarking Custom ASR Tools - Beyond The WER; episode five, Fine-Tuning ASR For Maximal Usability; and episode ten, How ASR Went From Frustration To ... Whisper Magic. This has been My Weird Prompts. If today was worth your commute, leave us a review on Spotify, and the whole archive lives at my weird prompts dot com.



Herman

See you tomorrow, brother.



Corn

If the room's quiet enough.

SOURCES

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

1. **NVIDIA Technical Blog, Entropy-Based Methods for Word-Level ASR Confidence Estimation . Retrieved 2026-11-08. [primary]**

   https://developer.nvidia.com/blog/entropy-based-methods-for-word-level-asr-confidence-estimation/

2. **NVIDIA Parakeet TDT 0.6B v2 model card, created 2025-04-15, last modified 2026-06-29, CC-BY-4.0 . Retrieved 2026-11-08. [primary]**

   https://huggingface.co/nvidia/parakeet-tdt-0.6b-v2

3. **OpenAI Whisper `timing.py`, `WordTiming` dataclass and `find_alignment`, showing `np.mean(text_token_probs[i:j])` word probability . Retrieved 2026-11-08. [primary]**

   https://raw.githubusercontent.com/openai/whisper/main/whisper/timing.py

4. **Whisper `transcribe()` signature and default thresholds . Retrieved 2026-11-08. [primary]**

   https://github.com/openai/whisper/blob/main/whisper/transcribe.py

5. **NeMo `GreedyCTCInfer` / `ConfidenceMethodMixin` source . Retrieved 2026-11-08. [primary]**

   https://docs.nvidia.com/nemo/speech/latest/_modules/nemo/collections/asr/parts/submodules/ctc_greedy_decoding.html

6. **NeMo Speech docs, Confidence Estimation YAML config . Retrieved 2026-11-08. [primary]**

   https://docs.nvidia.com/nemo/speech/latest/asr/asr_language_modeling_and_customization.html

7. **Greedy decoding confidence for CTC and RNNT, adds `frame_confidence`/`token_confidence`/`word_confidence` to `Hypothesis` . Retrieved 2026-11-08. [primary]**

   https://github.com/NVIDIA/NeMo/pull/4931

8. **Huo, Zhang & Tang, Identifying and Calibrating Overconfidence in Noisy Speech Recognition, arXiv:2509.07195v1, submitted 2025-09-08, accepted ASRU 2025 (full text read, pages 3**

   5). Retrieved 2026-11-08.
   https://arxiv.org/abs/2509.07195

9. **Jia & Van hamme, Leveraging Beam Search Information for Confidence Estimation in E2E ASR, arXiv:2607.29299v1, submitted 2026-07-31, ICASSP 2026 / IEEE OJSP. Retrieved 2026-11-08.**

   https://arxiv.org/abs/2607.29299v1

10. **Wang, Soltau, El Shafey et al., Word-level confidence estimation for RNN transducers, arXiv:2110.15222v1, submitted 2021-09-28, Proc. ASRU 2021. Retrieved 2026-11-08.**

    https://arxiv.org/abs/2110.15222v1

11. **An evaluation of word-level confidence estimation for end-to-end automatic speech recognition, arXiv:2101.05525 (full text read, first ~8,000 chars). Retrieved 2026-11-08.**

    https://arxiv.org/html/2101.05525v1

12. **Adopting Whisper for Confidence Estimation (C-Whisper), arXiv:2502.13446 (full text read, first ~8,000 chars). Retrieved 2026-11-08.**

    https://arxiv.org/abs/2502.13446

13. **Li, Zhang, Qiu et al., Improving Confidence Estimation on Out-of-Domain Data for End-to-End Speech Recognition, arXiv:2110.03327v2, submitted 2021-10-07, revised 2022-03-02, ICASSP 2022. Retrieved 2026-11-08.**
https://arxiv.org/abs/2110.03327v2

14. **Ravi, Raj T & Arora, TeLeS: Temporal Lexeme Similarity Score to Estimate Confidence in End-to-End ASR, arXiv:2401.03251v1, submitted 2024-01-06. Retrieved 2026-11-08.**
https://arxiv.org/abs/2401.03251v1

15. **Hikari-Tsai on `avg_logprob`/`compression_ratio`/`no_speech_prob` as decoding diagnostics. Retrieved 2026-11-08.**
https://github.com/SYSTRAN/faster-whisper/discussions/1042

16. **Enhancement Proposal: Add compression_ratio_hallucination_threshold... Retrieved 2026-11-08.**
https://github.com/openai/whisper/discussions/2420

17. **Bugfix: Illogical 'Avoid computing higher temperatures on no_speech' (comment on `no_speech_threshold` unreliability). Retrieved 2026-11-08.**
https://github.com/openai/whisper/pull/1903

18. **GNroy comment 2022-10-06 on per-frame word aggregation. Retrieved 2026-11-08.**
https://github.com/NVIDIA-NeMo/NeMo/issues/4661