Episode Book

EPISODE #5466

Hebrew Words Hidden in English Text

PUBLISHED	22 September 2026 at 15:08 UTC
DURATION	29:17
LISTEN	myweirdprompts.com/episode/hebrew-code-switching-lid/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
BOOK GENERATED	19 October 2026 at 17:36 UTC

SYNOPSIS

When Hebrew words are written in Latin characters inside otherwise English text, standard language identification fails: it returns the dominant label and the Hebrew vanishes. This episode works through the terminology (LID, CS-LID, transliteration, diacritization, G2P, text and script normalization), why romanized Hebrew defeats script-based heuristics, and what the literature actually offers — MaskLID, UniLID's tokenizer-based approach, STEN-TTS, DIVERS-Bench, and the EACL 2024 finding that no current approach is adequate. The gap is word-level, same-script, multi-label detection, and it sits precisely between two well-served categories.

TRANSCRIPT



Corn

...and that's the thing about the language tags, they're attached at the utterance level, so the moment you've got two scripts in one line of text, the whole thing collapses.



Herman

Which is exactly where our own pipeline keeps tripping over itself.



Corn

Right. Which is why Daniel sent us this one.



Herman

Daniel. Of course.



Corn

So Daniel's been poking at an improvement to the episode generation pipeline, and it's a three-stage thing. First, a classifier that looks at English text and picks out the words that are Hebrew but written in Latin characters. Second, a converter that takes those identified words and puts them into Hebrew script. Third, a script normalizer that adds the language tags so Chatterbox Multilingual, the TTS engine that runs this show, actually pronounces them correctly.



Herman

And he assumed this was a solved problem.



Corn

He did. His words, roughly: in a society where code-switching is part of everyday life for immigrants, he figured all of this would have been handled already. There are plenty of small Hebrew models for accessory needs. Diacritic restoration, adding the nekudot to



Corn (continued)

unvowelized Hebrew, that sort of thing exists and it's good.



Herman

But the classifier specifically.



Corn

Unless he's missing it, and he admits there's a chance he is, he couldn't find a classifier that looks at English containing Hebrew words and specifically classifies the Hebrew words. He doesn't think it would be that complicated a small model to build, and he thinks it would be of lasting value.



Herman

And then he goes wider.



Corn

He goes wider. He points out the surprising number of permutations you get once you start thinking about small language models bridging different languages. Language identification almost fulfills this need, except the catch is those languages are all in the same character set, and Hebrew isn't supported. There are niche models that transliterate Hebrew into transliterated English. There are plenty of world languages you can translate between that aren't written in the Latin alphabet. And if you take the general principle that the number of potential small model uses for Unicode conversion and transcription gets multiplied by the number of languages that don't share the English alphabet, and then by the permutations between them, you end up with an almost dizzyingly large pool of niche models.



Herman

The Spanish analogy.



Corn

Apply the same classifier to text with code-switched Spanish, classify the Spanish words, route them to Spanish-targeted TTS, and you get perfect pronunciation. Whereas with a single English TTS, you're hoping the model saw those Spanish loanwords in its training distribution.



Herman

Hoping is the operative word.



Corn

And his conclusion is that even though it looks niche, the potential uses are significant, which might be why there are institutions in Hebrew dedicated specifically to curating public utility and public interest models. He wants to get up to speed on the terminology, look at this class of small models, figure out which ones are most impactful and useful, how they're used, and how you tell whether what you need already exists or whether you're in blue sky territory.



Herman

That's a lot.



Corn

It's a lot. So where do we start?



Herman

Terminology, because Daniel asked for it and because the rest of the conversation is unusable without it. There are about seven terms that matter here, and they get conflated constantly.



Corn

Go.



Herman

Language identification, LID. That's the one everybody knows. You feed it text, it tells you what language the text is in. GlotLID, OpenLID, CLD3, the fastText-based models. Then code-switching LID, CS-LID, which is a different animal. That one tries to detect multiple languages within a single sentence. MaskLID is the reference implementation there.



Corn

Transliteration.



Herman

Mapping text from one script to another by sound. English to Hebrew name transliteration is the classic case. Then diacritization, or nikud restoration. That's adding the vowel marks to unvowelized Hebrew. Nakdan, Nakdimon, MenakBERT, D-Nikud, DiVRit. There's a small industry of these.



Corn

And the one that's actually the bottleneck for us.



Herman

Grapheme-to-phoneme. G2P. Converts written text into phonemes, usually IPA. Phonikud is the Hebrew one and it's good. Then text normalization, TN, which converts written forms into spoken equivalents for TTS. Numbers, dates, abbreviations, that whole class of problem. And finally script normalization, which unifies mixed-script text into one script.



Corn

Seven terms.



Herman

Seven. And the thing to notice is that none of them are the headline models. Nobody writes a press release about a diacritizer. These are accessory models. They're the plumbing.



Corn

The plumbing is the whole point, though. Our pipeline doesn't fail because the language model can't write. It fails because a Hebrew word written in Latin letters sails straight through the language tag and comes out of the speaker sounding like it's been run through a blender.



Herman

And that's the gap. The thing Daniel needs is word-level, same-script, multi-label. Word-level, because he needs to know which specific tokens are Hebrew. Same-script, because the Hebrew is written in Latin characters so there's no visual giveaway. Multi-label, because a single sentence can contain both English and Hebrew and both need tags.



Corn

And that combination falls between the cracks.



Herman

It falls between two very well-served categories. On one side you've got mainstream language identification, which is sentence-level and single-label. On the other side you've got Hebrew-specific tooling, which is excellent, but which assumes the input is already in Hebrew script. Neither one touches the middle.



Corn

Let's take the first one apart, because I think people assume LID is more capable than it is.



Herman

It's not a capability problem, it's a training-data problem. Sentence-level LID models are trained overwhelmingly on monolingual text. One document, one language, one label. So when you hand one a sentence that's mostly English with three Hebrew words embedded in it, it does exactly what it was trained to do. It returns the dominant label. English.



Corn

And the Hebrew words just...



Herman

Vanish. The MaskLID paper puts it plainly. In cases where a sentence is composed in both L1 and L2 languages, the LID classifier often only returns the dominant label L1. That's the exact failure Daniel is describing, and it's not a bug in any particular model. It's the shape of the task they were built for.



Corn

So the Hebrew words become noise.



Herman

They become English noise. Which is worse, because now they're not just untagged, they're actively mislabeled. And downstream, the TTS engine treats them as English orthography and pronounces them phonetically as English. Which is how you get a Hebrew word read out as though it were a surname from Ohio.



Corn

The character-set point is the one I want to sit on, because Daniel flagged it and I think it's the crux.



Herman

It's the crux. Most code-switching LID research targets either pairs that share a script or pairs that are script-distinguishable. Spanish-English shares the Latin script, so the model has to learn from context and phonotactics. Indonesian-English, same deal. Mandarin-English, the script is a giveaway, the characters tell you immediately. Arabic-English, same thing, Arabic script is unmistakable.



Corn

And romanized Hebrew?



Herman

Romanized Hebrew looks like ordinary English letters. There is no visual signal. "Shalom" is six Latin characters. "Mishpacha" is nine. A script-based heuristic has nothing to grab onto. The word is invisible to the exact class of feature that most CS-LID systems lean on.



Corn

So you need a model that's reading orthography and phonotactics, not script.



Herman

You need a model that has learned what Hebrew-in-Latin-letters looks like as a distribution. Which letters cluster, which digraphs show up, which vowel patterns are unusual in English. That's learnable, but it's not free, and it's not what the general-purpose tools were built to do.



Corn

And this is not a fringe complaint. There's a paper.



Herman

Burchell and colleagues, EACL 2024, "Code-Switched Language Identification is Harder Than You Think." Their conclusion is blunt. We present empirical evidence that no current approach is adequate.



Corn

That's about as clean a statement of the gap as you're going to get.



Herman

And what they do next is the interesting part. They reformulate CS-LID as a sentence-level multi-label tagging problem. Which is precisely the shape of the classifier Daniel wants to build. He's not asking for something weird. He's asking for the reformulation the field itself arrived at, applied to a language pair nobody's bothered with.



Corn

DIVERS-Bench.



Herman

September 2025. Existing models struggle to detect multiple languages within the same sentence, even as they score near-perfectly on curated monolingual data. So you get this bizarre situation where a model reports ninety-nine percent on the benchmark and then falls over the moment you hand it a realistic sentence from an actual bilingual speaker.



Corn

Which is a benchmark-validity problem as much as a modeling problem.



Herman

It's both. The benchmarks were monolingual, so the models optimized for monolingual. The real world isn't monolingual, so the models are useless in the real world. It's the oldest story in the field.



Corn

Okay. So what's the closest thing that exists?



Herman

MaskLID is the closest. It's training-free CS-LID. The trick is that it iteratively masks the features of the dominant language so the classifier underneath can find the second language. It runs on top of GlotLID or OpenLID, so you're not training anything, you're post-processing.



Corn

Sounds almost right.



Herman

It's almost right, and it's still not right. Two problems. First, it operates at sentence level. It'll tell you the sentence is mixed. It won't reliably tell you which word is which. Second, it relies on word-level language association scores to do the masking, and the 2026 follow-up work, Radoła and colleagues in the Findings of EMNLP, identifies that overreliance as a major flaw. The association scores are themselves noisy, so the masking inherits the noise.



Corn

So the closest existing approach is sentence-level and built on a shaky foundation.



Herman

That's a fair summary. It's a clever piece of work and it's not what Daniel needs.



Corn

What about the tokenizer angle? That one caught my eye.



Herman

UniLID. ICML 2026, "What Language is This? Ask Your Tokenizer." Instead of treating language identification as a classification problem over characters or words, they treat it as a tokenization problem. They use a UnigramLM tokenizer and the language falls out of how the text gets segmented.



Corn

And the efficiency numbers.



Herman

This is the part that matters for Daniel's plan. UniLID reaches sixty-nine percent accuracy with only five labeled samples per language, and eighty-nine percent with twenty-five.



Corn

Five samples.



Herman

Five. And it supports incremental language addition without retraining, which means you can bolt on a new language without rebuilding the whole thing. For somebody who's planning to generate a synthetic training set and train a small model, that's the most encouraging number in the literature. It says the data requirement is modest.



Corn

Modest, but not zero.



Herman

Not zero. There's still a real annotation question. But five samples per language is a weekend of work, not a research program.



Corn

And then there's the one that's basically the architecture Daniel described, just for a different language pair.



Herman

STEN-TTS. Indonesian-English, O-COCOSDA 2024. They took a multilingual TTS and inserted a finetuned BERT for per-word language identification into the text-to-phoneme conversion stage.



Corn

Which is the classifier, in the pipeline, doing exactly the job.



Herman

Exactly the job. And the result is the line that should be printed on a poster somewhere. The code-switching model achieves superior naturalness and improved speech intelligibility compared to the Indonesian and English baseline.



Corn

So the thing Daniel wants to build has already been built once, in Indonesian.



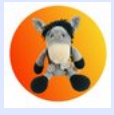
Herman

The architecture has been validated. The Hebrew instantiation doesn't exist. That's the honest state of play. Somebody proved it works, and then nobody generalized it.



Corn

Which is a strange thing to sit with, because the generalization is not obviously hard.



Herman

It's not obviously hard. It's just that nobody's incentive pointed at it. Indonesian-English is a big enough pair that a research group would take it on. Hebrew-in-Latin-characters is small enough that it falls below the threshold where anybody bothers.



Corn

Let's move to the Hebrew ecosystem, because that's where Daniel's institutional observation comes in and I think he's right about it.



Herman

He's right about it. Dicta is the anchor. If you're doing anything with Hebrew NLP and you haven't looked at Dicta, you're working harder than you need to.



Corn

Nakdan.



Herman

Nakdan is their diacritizer, ACL 2020. It combines neural models with curated linguistic knowledge, which is the right architecture for Hebrew specifically, because Hebrew diacritization has genuine ambiguity that pure statistical models handle badly. It supports Modern, Rabbinic, and Poetic Hebrew, and it's freely accessible. Nakdan Pro, you can just go use it.



Corn

Which is remarkable, actually. A professional-grade diacritizer, free, for a language with maybe nine million speakers.



Herman

That's the institutional point. Somebody decided this was public infrastructure and funded it accordingly.



Corn

And Dicta-LM 3.0.



Herman

February 2026. Open-weight Hebrew language models in twenty-four billion, twelve billion, and one point seven billion parameters. Adapted from Mistral Small, NVIDIA Nemotron Nano, and Qwen3 respectively. Sixty-five thousand token context. And they shipped a Hebrew benchmark suite alongside it covering translation, summarization, Winograd, Israeli trivia, and diacritization.



Corn

The framing in that paper is the part I want to quote.



Herman

Sovereign large language models. That's their phrase. Remain low in supply yet high in demand. And then the line that generalizes the whole thing. They describe it as a framework that can be leveraged for adapting other LLMs to various non-English languages.



Corn

Which is the template. That's what a language community does when it decides it wants its own models rather than renting somebody else's.



Herman

It's a blueprint. And it's an argument that the Hebrew case isn't special. Any language community with a few competent people and a modest budget can do this now.



Corn

Let's run the small models, because Daniel specifically said there are plenty of these for accessory needs and he's right.



Herman

Nakdimon, NAACL 2022. Two-layer character-level LSTM, and the notable thing is it diacritizes without any human-curated resources. No lexicon, no rules, just the model. MenakBERT, 2024, character-level transformer. And it turns out diacritization finetuning transfers to part-of-speech tagging, which is a nice result about how much morphological information the diacritics carry.



Corn

D-Nikud.



Herman

LSTM plus TavBERT hybrid, state of the art on modern texts. Then DiVRit, which is the one I'd point at if you want to see how creative this space gets. They frame diacritization as zero-shot classification using a Hebrew visual language model.



Corn

Visual.



Herman

They render the diacritized candidates as images and let the vision model pick. Which sounds absurd until you realize that diacritics are, physically, small marks positioned relative to letters. Rendering them and looking at them is not a crazy way to encode the problem.



Corn

It's using the vision stack as a spatial reasoning engine.



Herman

That's exactly what it is. And it works well enough to be worth publishing.



Corn

Phonikud.



Herman

Phonikud is the one that matters most for Daniel's pipeline, because it's the G2P stage. Open-source Hebrew grapheme-to-phoneme, outputs fully-specified IPA, and it comes with the ILSpeech corpus and a benchmark. Interspeech 2026.



Corn

And the finding.



Herman

Small, local TTS models with phonetic input from Phonikud approach large proprietary systems.



Corn

Say that again, because that's the thesis of the whole episode.



Herman

Small, local TTS models with phonetic input from Phonikud approach large proprietary systems. You don't need a giant model. You need a small model and the right phonetic input. The intelligence lives in the accessory, not in the headline.



Corn

And ReNikud.



Herman

ReNikud is audio-supervised. It uses ASR pseudo-labeling on thousands of hours of unlabeled audio, so it learns natural spoken norms rather than formal grammar. Which is a different target. Formal diacritization and how people actually talk are not the same thing.



Corn

Let's do the permutation argument, because Daniel spent real estate on it and I think it holds up better than it sounds.



Herman

It holds up. Transliteration alone spans a remarkable range. Cohen and Elhadad did English to Hebrew name transliteration, and the interesting wrinkle there is multicultural name origins. You're not just mapping sounds, you're guessing which language the name came from before you can transliterate it. Cheng and colleagues did English to Chinese, Hebrew, and Thai. And then there's Judeo-Arabic to Arabic script, EACL 2026, where Hebrew-script Arabic has to be transliterated and the text contains frequent code-switching into Hebrew.



Corn

So you're transliterating a text that's already code-switched, into a script that isn't the one it's written in.



Herman

Three layers of transformation on a single document. And that's one language community.



Corn

The Rom and Bar result is the one that made me sit up.



Herman

Rom and Bar, 2024. They trained a bilingual Arabic-Hebrew model by transliterating Arabic into Hebrew script to force a shared character space. And it outperformed the model that kept Arabic in Arabic script.



Corn

Despite less data.



Herman

Despite sixty percent less training data. Which is a striking result. It says script handling is not cosmetic. It's not a preprocessing detail you do for convenience. Changing the script changes what the model can learn.



Corn

Why?



Herman

My read is that a shared character space lets the model share subword statistics across both languages. If Arabic and Hebrew are in different scripts, the tokenizer produces disjoint vocabularies and nothing transfers. Force them into one script and suddenly the morphological patterns, the root structures, the affix behavior, all of it can be learned jointly. Semitic languages share a lot of structure and the script was hiding it.



Corn

Which is a nice inversion of the usual intuition. Normally you'd say preserving the original script preserves information.



Herman

You'd think so. But for a model, the original script is a barrier, not a signal. The information is in the morphology, and the script is just an encoding that happens to differ.



Corn

Let's do the Spanish analogy properly, because I think it's the most useful thing in Daniel's prompt.



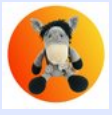
Herman

It's the most useful thing because it's the general principle stated in a case everybody can picture. Take a sentence in English with Spanish words in it. Run the classifier. Tag the Spanish words. Route them to a Spanish-targeted TTS voice. Now the pronunciation is correct by construction.



Corn

Instead of by hope.



Herman

And the failure mode of hope is specific. A single multilingual TTS model has seen some Spanish loanwords in its training distribution. It has not seen all of them. It has not seen them in every morphological form. And it has no way to tell you which ones it's confident about and which ones it's guessing.



Corn

So you get inconsistent pronunciation within a single episode.



Herman

You get inconsistent pronunciation within a single paragraph. And that's the worst outcome, because inconsistency is more jarring than uniform error. A listener can adapt to an accent. They can't adapt to a word that's right in one sentence and wrong in the next.



Corn

And the STEN-TTS result is the proof that routing works.



Herman

It's the proof. Per-word language identification in the pipeline, measurable gains in naturalness and intelligibility over the single multilingual baseline. The routing converts a probabilistic hope into a deterministic pipeline stage.



Corn

Which is a general principle well beyond TTS.



Herman

It's a general principle about pipelines. When you're relying on a model to have absorbed something, you're relying on a distribution. When you route explicitly, you're relying on a decision. Decisions are testable. Distributions are not.



Corn

Right. So let's give the practical answer, because Daniel explicitly asked how you tell whether what you need exists or whether you're in blue sky.



Herman

Four steps. First, check sentence-level LID. GlotLID, OpenLID, CLD3, UniLID. Cheap, fast, and it'll handle the easy cases. But expect it to fail on same-script code-switching, and don't be surprised when it does.



Corn

Second.



Herman

Check CS-LID methods. MaskLID, and the ILP-refined version. These are good at telling you that a sentence is mixed. They're weaker at telling you which word is which. If you need per-word labels, they won't get you all the way.



Corn

Third.



Herman

Check the language-specific institution. For Hebrew that's Dicta. And the important thing here is that they may have the component even if it's not packaged as a standalone classifier. A lot of institutional NLP work ships as a service or as part of a larger tool, and people miss it because they're searching for a Hugging Face model card.



Corn

Fourth.



Herman

If the task is word-level, same-script, and low-resource, assume blue sky. Those three properties together are the gap. And then note that UniLID's few-shot numbers say a synthetic-data approach is viable with modest annotation. Sixty-nine percent with five samples, eighty-nine with twenty-five.



Corn

Which turns "blue sky" into "a few weekends."



Herman

It turns it into a project. Not a research program.



Corn

I want to push on that, because I think there's a trap in the synthetic-data plan.



Herman

Go ahead.



Herman

I mean, I have a suspicion about it, and I'd rather hear yours first.



Corn

My suspicion is that synthetic data teaches the model the distribution of the generator, not the distribution of the world. If you generate your Hebrew-in-Latin examples with a script, you're going to get clean, consistent transliterations. And real Hebrish is not clean. People spell the same word three different ways in one paragraph.



Herman

That's a real concern and I don't have a clean answer to it. The honest version is that synthetic data gets you the easy eighty percent and the last twenty is where the actual work lives. You'd want some real-world data in the mix, even a small amount, to anchor the spelling variation.



Corn

Which is the same lesson as DIVERS-Bench, just one level up. Benchmarks that don't reflect real usage produce models that don't work in real usage.



Herman

And synthetic training data is a benchmark you're also training on. If it's too clean, you've built a model that's excellent at a world that doesn't exist.



Corn

I think that's the sharpest thing we've said.



Herman

I'll take it.



Corn

One more thing before we wrap the discussion, which is the institutional question. Daniel said it might be why there are institutions in Hebrew dedicated to curating public utility models. I think he's got the causality slightly backwards.



Herman

How so?



Corn

I don't think the institutions exist because the tasks are niche. I think the tasks are solved because the institutions exist. Somebody decided Hebrew was worth public infrastructure, and then the diacritizers and the G2P models and the language models followed from that decision. The causality runs from the funding to the tooling, not the other way around.



Herman

That's right, and it explains the gap Daniel found. Hebrew-in-Latin-characters falls outside the institutional mandate in a subtle way. Dicta's mandate is Hebrew. Hebrew written in Latin characters is, from a certain institutional perspective, not Hebrew. It's English with foreign words in it. So it falls between the Hebrew institution and the English-language tooling, and nobody owns it.



Corn

Which is a specific and fixable kind of gap.



Herman

It's the most fixable kind. It's not a hard technical problem. It's an ownership problem.



Hilbert

He's right, and it's not new.



Herman

Sorry, go on.



Hilbert

I worked a summer at a Jewish newspaper in New York. Back office, not the newsroom. They handed me boxes of old letters to the editor and told me to digitize them.



Corn

How many boxes?



Hilbert

Forty-one. The letters were in English, mostly, but a lot of them were from immigrants and they'd write the Hebrew words in Latin letters. Shabbos. Mishpacha. Tzedakah. And the OCR software flagged every one of them as a spelling error.



Corn

So your job was to sit there and fix them.



Hilbert

Eight hours a day. One at a time. And after about a week I stopped thinking about it and started just knowing. You develop a checklist without noticing you're developing it. Does the word have a ch or a tz that doesn't fit English? Does it end in a vowel English words don't usually end in? Is it sitting next to other Yiddish or Hebrew loanwords?



Herman

So you were running a classifier.



Hilbert

I was running a classifier. And I could probably still do it today. What I'm not sure about is whether a model learns it from synthetic data. Because the thing I was actually using wasn't the spelling. It was knowing who wrote the letter.



Corn

The context.



Hilbert

A woman writing from the Bronx in the fifties about her synagogue is going to use Hebrew words a certain way. A man writing from the Lower East Side is going to use them a different way. I don't know how you put that in a training set.



Herman

You'd need the metadata.



Hilbert

And nobody kept it. The letters were in a box.



Corn

Did you keep any of them?



Hilbert

I kept one. A letter from a woman in the Bronx complaining about the synagogue's new rabbi. She wrote shul three different ways in the same paragraph. S-H-U-L, S-H-O-O-L, and S-C-H-U-L. I've always wondered if that was a mistake or just how she thought about it.



Herman

Probably how she thought about it.



Hilbert

That's what I figure. Anyway. Somebody's waiting for me in the car park.



Corn

How long?



Hilbert

A while.



Corn

Okay. So the thing I keep coming back to is that the gap Daniel found isn't a technical gap. It's an ownership gap. Hebrew-in-Latin-characters falls between the Hebrew institution and the English-language tooling, and neither one considers it theirs.



Herman

And the fix is somebody deciding it's theirs.



Corn

Which is what Daniel's proposing to do, essentially. Build the thing because nobody else will.



Herman

The one thing I'd leave people with is the Phonikud result, because it generalizes. Small, local TTS models with phonetic input approach large proprietary systems. The intelligence lives in the accessory model, not the headline model. Every pipeline that's struggling with a language edge case is probably struggling because it's missing an accessory, not because it needs a bigger model.



Corn

And if you're building something in this space, or you've found a model that already solves this problem, let us know. We'd like to hear about it.



Herman

Thanks as always to Hilbert Flumingtop, our producer.



Corn

This has been My Weird Prompts. If you want to get in touch, email us at show at my weird prompts dot com.



Herman

We'll be back soon.



Corn

See you tomorrow.