Episode Book

EPISODE #6161

Turning Words Into Sound: G2P, IPA, and the Hebrew Vowel Problem

PUBLISHED	20 October 2026 at 14:01 UTC
DURATION	24:30
LISTEN	myweirdprompts.com/episode/g2p-ipa-hebrew-pronunciation-06510/
HOSTS	Corn, Herman
SCRIPT MODEL	anthropic/claude-sonnet-5.5
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.10
BOOK GENERATED	21 October 2026 at 07:50 UTC

SYNOPSIS

Daniel records himself reading prompts aloud to fine-tune a speech recognition model, and every brand name, Hebrew term, or Yiddish word he's only seen written down becomes a wall: he doesn't know how to say it, so whatever he says becomes the training label. The obvious fix is a pipeline — word in, predicted phonetics, a voice says it back. The pipeline exists, in parts. Grapheme-to-phoneme (G2P) conversion, IPA versus ARPAbet, dictionary lookup before neural guessing, WikiPron's five million human-written entries, and Hebrew's missing vowels all shape what's actually possible. This episode walks through the tools — espeak-ng, Epitran, Charsiu G2P, Phonikud, ReNikud — and the playback half nobody warns you about: many modern TTS models ignore phoneme input entirely. If you've ever needed a machine to tell you how to say something, this is the map.

TRANSCRIPT



Corn

The standard answer to "how do I hear a word before I say it" is that you need an IPA model. Text goes in, the phonetic alphabet comes out, you pipe that into a voice, done. Every tutorial lands there. And the catch is that IPA is not a model. It's a notation. It's the sheet music, and nobody has told you yet who's playing it.



Herman

Nobody has, and the player is worst at exactly the words you'd care about. But I'm getting ahead of the brief. You've got the setup, Corn.



Corn

Daniel records himself reading prompts aloud, hour after hour, to fine-tune a speech recognition model. Parakeet mostly, the NVIDIA TDT six hundred million parameter one, version two, and sometimes Whisper. The recordings are the ground truth, so when a prompt has a brand name, a Hebrew or Yiddish term, a place name he's only ever seen written down, he hits a wall. He doesn't know how to say it, and whatever he says becomes the label for the training clip. The capture app he has has a pronunciation guide section, and that made him wonder whether there's a model that turns a word into IPA, in any language, and whether there's text to speech that accepts IPA as input. His imagined pipeline is word in, predicted phonetics behind the scenes, a voice says it, he listens, then he records.



Herman

The pipeline exists, in parts, which is the good news. The field calls text to phonetics grapheme to phoneme conversion, G to P. Graphemes are the letters, phonemes are the sounds. If you search for "text to IPA model" you find some of it. If you search G to P you find all of it.



Corn

And the bad news is the parts don't fit together as cleanly as the diagram.



Herman

They fit, with gaps. Let me lay out the vocabulary, because it decides which tools you can use. IPA, the International Phonetic Alphabet, is the universal notation, one symbol per sound across all languages. ARPAbet is the older American English only system, built from plain capital letters, with a digit on each vowel for stress. So the sound in "fire" shows up as something like AY, then a one, then ER. It's what the CMU Pronouncing Dictionary uses, which is a hand-built lexicon of well over a hundred thousand English words.



Corn

Hold on, I want to know where the dictionary sits in this. Because if there's a dictionary, why would anyone need a model?



Herman

That's the correct instinct, and it's the architecture almost every G to P tool shares. You look the word up first. If it's in the lexicon, you use the human-checked pronunciation and you're done. Only when the lookup misses do you fall back to a neural model that guesses from spelling patterns. The popular G to P en package works exactly this way for English: dictionary lookup, then a neural guesser for anything out of vocabulary, and it outputs ARPAbet, not IPA.



Corn

So the model is the spare tyre.



Herman

The spare tyre, yes. And Daniel's words are the ones that blow the first tyre. Names, brands, loanwords. Those are by definition the things not in the dictionary, so the guesser is doing all the work, and a guesser trained on common English spelling patterns will apply those patterns to a word that doesn't follow them.



Corn

Give me a case where that bites, because right now it sounds abstract.



Herman

Any brand name made by squashing two words together, or spelled for the logo. The model sees letters and has no way to know the founder pronounces the middle bit differently from how English would. It's not that it fails loudly. It produces a clean, confident, perfectly formed phonetic string that is wrong. And then, in Daniel's pipeline, a voice reads that string back to him in a calm, authoritative tone.



Corn

So I learn to pronounce it wrong, with the voice vouching for me.



Herman

That's the risk I'd design around. We'll come back to it with a confidence flag. First the tools, because there are more than people expect. Espeak-ng is the workhorse. It's a rule-based synthesiser that carries pronunciation rules for something like a hundred and twenty-seven languages and accents, and Hebrew is on the list. A Python wrapper called phonemizer sits on top of it and is how most people get IPA out of it. I'll flag that the exact command line switch for IPA output is from memory on my side, so check the docs. Also phonemizer is GPL licensed, which matters if Daniel ever ships the app.



Corn

He's shipping it to himself. He can probably live with the GPL.



Herman

Fair, personal use is a different world. Then there's Epitran, also rule-based, roughly a hundred and fifty language and script pairs. The thing to know about it is that rule-based means it works when the spelling is regular, Italian or Spanish or Turkish, and it struggles when spelling is a loose suggestion, English being the extreme case. Hebrew isn't in its table at all.



Corn

Rule-based, so English is its natural enemy. I can see why nobody leads with it.



Herman

Now the one closest to what Daniel described: a neural, multilingual, any-language model. Charsiu G to P. It's a byte-level T-five model, covers about a hundred languages, MIT licensed. The trick is you have to tell it the language for every word, with a tag in front, so you say this word is American English or this word is Italian. And its best published model gets about a quarter of whole words wrong on the benchmark, a word error rate around twenty-six percent. The authors themselves call it beta, not camera ready.



Corn

Hang on. A quarter of words wrong. That's on normal words?



Herman

On the benchmark, which is dictionary-style words in many languages. On names and brands, my expectation is that it's worse, but I'm saying that as an expectation. I haven't seen anyone measure it on names.



Corn

That's the part I find strange. You'd think someone would have asked a large language model to write IPA for a list of surnames and scored it.



Herman

People have tested language models on phonetics. There's a benchmark where models write IPA for common American English words. Not names. There's a Japanese study that flags proper nouns as the main source of errors. And a paper on Persian found prompted language models beat the traditional G to P tools without any training. So the direction is plausible, but for Daniel's case I can't point at a number. My guess is a large model does better than a spelling-pattern guesser on brands, because it may have seen the company talk about itself, and worse on things it has never seen. Corn, would you trust it?



Corn

For a brand it has read ten thousand times, yes. For a Hebrew street name in a neighbourhood nobody wrote about, I'd trust it the way I trust a stranger giving directions. Warmly, and I'd check a map.



Herman

That's a good rule. Familiarity decides it. Which is why I'd put the lookup first, and there's a good lookup beyond the English dictionary. WikiPron is a collection scraped from Wiktionary, more than five million word and pronunciation pairs across three hundred and forty-eight languages, already in IPA, downloadable as plain tab-separated files. A human wrote those entries. If the word is there, you skip the guessing.



Corn

Which covers "Worcestershire" and doesn't cover the artisanal cheese shop.



Herman

Right, and the cheese shop is also unlikely to be a word Daniel needs often. The dictionary catches the long tail of the famous, and the model gets the rest.



Corn

Then Hebrew. Daniel's prompts have it, and Yiddish, and he lives in Jerusalem, so that's not an exotic case. Espeak supports Hebrew, you said. So where's the problem?



Herman

The problem is the writing system. Ordinary Hebrew text leaves out most of the vowels. A written word can be read several ways, and you work out which from context and from knowing the language. The dotted marks, nikud, that fix the vowels exist, but nobody writes with them day to day. So a tool that converts spelling to sound has to choose a vowel pattern the text doesn't contain.



Corn

And English speakers do the same trick with "bt" in a text message, so I'm not going to pretend that's strange.



Herman

Except you have context and a fluent speaker's ear. A G to P tool doesn't. There are two recent projects aimed straight at this, by the same developer. Phonikud takes Hebrew with the vowel marks and gives you IPA. It comes with a diacritiser, a small model that adds the marks to plain text, plus extra marks for where the stress falls and for the vocal shva. Its example turns the Hebrew for "hello world" into a clean IPA string with stress marked on each word. The second is ReNikud, newer, which skips the middle step. Plain Hebrew without marks goes in, IPA comes out, in a model of about twenty megabytes, and anything that isn't Hebrew passes through unchanged. The Phonikud page itself says ReNikud is the higher accuracy option.



Corn

Passes through unchanged. So if a prompt has a Hebrew word embedded in an English sentence, the English stays English.



Herman

Which is how you'd want it. Now, the caveat, and this is from the project's own notes: names and non-Hebrew words are sometimes wrong. I haven't got accuracy figures for either tool, the repositories don't publish any, and I haven't opened the papers. So I can tell you they exist, they target the right problem, and they're honest about the limit that matters for Daniel.



Corn

Which is names. Again.



Herman

Again. Names, in a language where the vowels aren't written, are the hardest case there is. And Yiddish is a separate matter. It's written in Hebrew letters but it's a different language with its own pronunciation. I haven't seen a dedicated G to P tool for it, and I'd be inventing if I told you one exists. The WikiPron route might cover some words. For the rest, I'd say you want a human or a recording.



Corn

Which, for a one-man dataset, is also fine. He only needs one person who speaks it.



Herman

True. But let's get to the playback half, because that's where Daniel's idea gets bumpy. Suppose you've got IPA for the word. What takes IPA as input?



Corn

I'd have guessed all of them. Isn't a phoneme the thing a voice is built from?



Herman

That's the intuition, and it's right for older designs. A classical synthesiser, or a model like VITS, has a front end that turns text into phonemes, and the voice model consumes phonemes. So you can swap in your own. The newer large models are different. Many of the modern, language-model-style and audio codec voices read text directly, learned end to end from text and audio pairs, and they either ignore phoneme tags or skip them. So you can send a perfect IPA string to a fashionable model and it reads the letters or gets confused.



Corn

So the newest option is the least controllable. That tracks with most of technology.



Herman

It does. So the short list. The cloud services are the reliable route. Azure's speech service has a phoneme element in its markup, you give it an alphabet, IPA being one choice, and the phonemes. If you send a symbol it doesn't recognise, you get an error back rather than a guess, which I like. One quirk: if you mark stress in IPA, it wants you to mark it on every syllable. They also support custom lexicons, the pronunciation lexicon standard, up to a hundred kilobytes, case sensitive.



Corn

A lexicon he'd build himself, over time, as his word list grows.



Herman

That's attractive for him. Google Cloud's text to speech has a similar tag, one word per tag, IPA or X-SAMPA. The page doesn't say which voice types honour it, so test the one you intend to use. Amazon Polly has per-language tables of IPA symbols. I couldn't confirm which voices or engines support the tag, and I couldn't confirm whether Polly has a Hebrew voice at all. Same for Azure and Google on Hebrew phoneme tags: unchecked.



Corn

So Hebrew preview from the cloud is a coin toss until he tries it.



Herman

A coin toss I can't call, yes. ElevenLabs is the next one. Phoneme tags work only on one of their older fast models, the flash version two, and they recommend the CMU Arpabet alphabet over IPA. On their newest model you can put IPA between slashes in the plain text. Other models want respelling, spelling the word the way it sounds, or an alias tag that swaps the written word for another word.



Corn

Spelling it the way it sounds is just what I do when I text my brother a name I can't say. It can't be the professional solution.



Herman

It's more professional than it sounds, and I'll get there. Then the local options, which matter if Daniel wants this offline and free. Kokoro, the eighty-two million parameter open model, Apache licensed. It has a documented inline override: you write the word in markdown link style, and between slashes you give the phonemes you want. One caution. Kokoro uses its own phoneme set, from its companion library Misaki, which isn't plain IPA. It uses a capital letter for certain diphthongs, for example. And I haven't confirmed whether there's a documented way to pass a whole raw phoneme string to it. The README doesn't describe one, and I'd have to read the source. Also, Kokoro's language list has no Hebrew.



Corn

So for Hebrew words in an English sentence, Kokoro would read the Hebrew in an English accent.



Herman

Unless you supply phonemes close to Hebrew sounds, using the symbols its English voice understands. It gets you in the neighbourhood. Then Piper, a small fast local engine used a lot in home automation. Its models are trained on phoneme sequences from espeak-ng, so the machinery is phoneme-based. The original repo was archived last October and development moved to a fork that embeds espeak-ng. Whether it has a clean syntax for feeding raw phonemes, I could not confirm, so I won't pretend.



Corn

You've said "unconfirmed" so many times I'm starting to feel I should sign something.



Herman

It's a topic where the documentation is thin, and I'd rather tell you that than invent a flag name. The structure holds up though: cloud services accept IPA reliably, small local engines accept it through their phonemiser, and the big end-to-end models often don't.



Corn

Then let me argue the other side for a second. Why does Daniel need IPA at all in the loop? If a modern voice can say most words reasonably from spelling, he could just type the word and hear what the voice does.



Herman

For a word the voice model knows, that works, and often better than a hand-built phoneme string. The case for IPA is control. When the preview says it wrong, with spelling alone you can't fix it. With phonemes, you can say exactly what you meant, listen, and adjust. And there's the Hebrew point: an English voice reading Hebrew letters does something strange. You need the phoneme layer to get it to produce Hebrew-ish sounds at all.



Corn

So IPA is the escape hatch for when the spelling route fails.



Herman

And that connects to the human side, which is the question of what Daniel sees. His idea was to hide the IPA. I think that's right, because IPA is a poor interface for a person who isn't a phonetician. Does he read it?



Corn

I'd guess no. He'd see a string of symbols and trust it blindly, which is worse than seeing nothing. Why, do you read it?



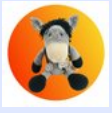
Herman

Slowly, and with mistakes. That's why the guide section in his app should show a respelling. Plain letters, hyphens between syllables, the stressed syllable in capitals. Wikipedia's convention is a decent template: ordinary English letters, one special symbol for the unstressed vowel, and the whole thing designed so an English reader gets it first time. Something like pruh-NUN-see-AY-shun for "pronunciation".



Corn

And that's what I'd put on a sticky note by the microphone.



Herman

Right. Two rules from that same page matter here. One, don't use respelling for non-English words. It assumes English sound values, so for a Hebrew word it misleads. Two, avoid it when a syllable spells out another word, like Maui turning into "wow-ee". For Daniel's Hebrew and Yiddish terms, the guide needs the audio and the IPA, not a respelling.



Corn

That's a reasonable split. Respelling for the English brands, the sound file for everything foreign.



Herman

Now the ASR half, because the second part of Daniel's question was how IPA integrates with speech to text. And the answer is it mostly doesn't. Neither Parakeet nor Whisper uses phonemes. Both are trained on audio paired with text. The Parakeet card describes a FastConformer encoder and a TDT decoder, and the output is text with punctuation and capitals. Its tokenizer was built from the training transcripts. I inferred from the file names that it's a byte-pair style tokenizer, so call that likely rather than confirmed. Whisper was trained weakly supervised on around six hundred and eighty thousand hours to predict transcripts. I believe its tokenizer is byte-level, but I haven't verified that on a page.



Corn

So IPA never touches the training data.



Herman

Never. The label is the spelling. The model hears whatever Daniel says and learns to map that sound to those letters. So the preview pipeline is purely a pronunciation aid for him. It changes what he records, not what the model is trained on in format.



Corn

Which is where I'd worry. If the aid makes him say "Kubernetes" one way for a month and then he changes his mind, the dataset has both.



Herman

And for a personal model that's fine, it's the point. The model should learn Daniel's voice, with his pronunciations. But a dataset card should say so. It should say these are one speaker's choices, this accent, and where he deliberately used a particular pronunciation of a name or a loanword, so anyone using the data later knows the sound isn't a reference pronunciation. I'd put in the card that a few terms were rehearsed with a synthetic preview. I'm not certain how much that matters, I'd rather say it than leave it out.



Corn

Herman, what about the reverse direction? He's recorded the take. Could he check he said what he intended?



Herman

Yes, and it's the best use of the IPA he's already produced. There are phone recognisers, models that output phonetic symbols instead of words. A Facebook wave to vec two model fine-tuned on Common Voice outputs strings in the espeak phoneme set, multilingual, zero-shot, Apache licensed, expects sixteen kilohertz audio. And a newer system called ZIPA, from an ACL paper this year, trained on roughly seventeen thousand hours across many languages, outputs IPA directly. Its authors say it's still weak on sociophonetic variation, meaning accent differences.



Corn

Accent differences are exactly what Daniel is. An Irishman in Jerusalem saying Hebrew words.



Herman

Which means don't expect the check to agree with the target string exactly. You'd compare how close it is, and look for a big gap, not a perfect match. If the target is a stressed second syllable and the recogniser heard the first, flag the clip.



Corn

I like that, and I think it's the only part I'd actually trust to save time. A flag saying "this one sounds different from your plan" is useful even if the recogniser is a bit wrong.



Herman

So let me put the pipeline together, and Corn, interrupt me. First, a lookup against a pronunciation dictionary and the WikiPron data, plus a personal lexicon of everything Daniel has already confirmed. That's step one because a hit is a human's work. Second, on a miss, G to P with a language tag, which means the app needs to know the word's language. Hebrew words go to ReNikud or Phonikud, English to a neural English model or espeak, and Charsiu G to P or espeak for others. Third, the preview through an engine that takes phonemes: a cloud service if he doesn't mind sending words out, or a local one. Fourth, a respelling displayed to him for English words.



Corn

And the fifth is the flag, because I don't want a confident wrong answer.



Herman

Yes, the confidence flag. A dictionary hit is marked trusted. A G to P guess is marked unverified, in a different colour, with the message that the voice is guessing. And I'd go further. When it's unverified, the app could play two or three variants, a different stress, a different vowel, and ask him which one the word actually is.



Corn

And when none of them are right, he asks a person and types the answer in, and it goes into the personal lexicon so it's a trusted hit next time. So the mistakes pay for themselves.



Herman

That's the loop. The longer he uses it, the fewer guesses he makes. For a recurring name or term, a year in, almost every word is a hit.



Corn

One more question. Could the language model he's already using do step two? Ask it to give the phonemes with a language, and let it say when it doesn't know?



Herman

You could try, and I'd say try it as an extra voter rather than the only one. If the language model and the G to P tool agree, the confidence goes up. If they disagree, flag it. I'm speculating there, since nobody's published numbers for it on names.



Corn

A disagreement detector. I'd buy that for a lot of things in my life.



Herman

And that is a fair summary of the design, even if I'd avoid summarising it any harder. Lookup wins, guesses get flagged, Hebrew gets its own path, the preview needs an engine that accepts phonemes, and the dataset card says the labels are text and the sound is Daniel's.



Corn

This has been My Weird Prompts. Thanks to our producer, Hilbert Flumingtop, for putting the episode together. If you want more, find us at my weird prompts dot com.



Herman

See you tomorrow.