The General Pipeline for Processing Podcast Data

IC²S² 2026 Tutorial

David Jurgens and Dallas Card
July 28, 2026

How To Go From a URL To a Dataset?

```
RSS feed → audio file → transcript → speaker turns → roles → features → table
```

- Each arrow is a measurement decision, and by the last arrow you are modeling the pipeline's artifacts as much as the podcast.
- Every arrow is a **model**, and every model has an **error rate**
- Errors compound *asymmetrically*

The General Pipeline Architecture

- **Transcription and diarization are independent models.** Neither sees the other's output

Stage	Input	Output	Standard tools
Acquire	RSS URL	audio file + metadata	`feedparser`, `requests`, `yt-dlp`
Normalize	mp3/m4a	16 kHz mono WAV	`ffmpeg`
Transcribe	audio	text + segment times	faster-whisper, Parakeet
Diarize	audio	*who spoke when*	pyannote 3.1
Align	audio + text	**word**-level times	whisperX (wav2vec2 CTC), MFA
Assign roles	turns + metadata	host / guest	heuristics + LLM
Extract features	audio + turns	F0, MFCC, rate, overlap	openSMILE / eGeMAPS

`feedparser` — https://feedparser.readthedocs.io/, `yt-dlp` — https://github.com/yt-dlp/yt-dlp, `ffmpeg` — https://ffmpeg.org/

faster-whisper (CTranslate2) — https://github.com/SYSTRAN/faster-whisper

pyannote speaker-diarization-3.1 — https://huggingface.co/pyannote/speaker-diarization-3.1

Bain, Max, Jaesung Huh, Tengda Han, and Andrew Zisserman. "WhisperX: Time-Accurate Speech Transcription of Long-Form Audio." *INTERSPEECH 2023*. arXiv:2303.00747. https://arxiv.org/abs/2303.00747

McAuliffe, Michael, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. "Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi." *Interspeech 2017*, 498–502. https://doi.org/10.21437/Interspeech.2017-1386

Eyben, Florian, Klaus R. Scherer, Björn W. Schuller, Johan Sundberg, Elisabeth André, Carlos Busso, et al. "The Geneva Minimalistic Acoustic Parameter Set (GeMAPS) for Voice Research and Affective Computing." *IEEE Transactions on Affective Computing* 7(2), 2016, 190–202. https://sail.usc.edu/publications/files/eyben-preprinttaffc-2015.pdf

What makes this data possible now?

- **~2010s** — Kaldi. HMM/GMM then DNN hybrids. Required a pronunciation lexicon and an in-domain language model. Still 15.5–18.2% WER on the standard conversational benchmark
- **2019–20** — wav2vec / wav2vec 2.0: self-supervised pretraining on unlabelled audio
- **Sep 2022** — **Whisper**. 680,000 hours of weakly supervised multilingual data, trained as plain sequence-to-sequence. Robust *zero-shot*, with no lexicon and no tuning
- **2023–24** — the efficiency wave: CTranslate2 (faster-whisper), GGML (whisper.cpp), distillation, CoreML/Metal backends
- **2024–25** — NVIDIA Parakeet / Canary: better and much faster, via ONNX even on CPU

Accurate podcast transcription is now feasible at scale

Radford, Alec, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. "Robust Speech Recognition via Large-Scale Weak Supervision." arXiv:2212.04356, 6 December 2022; published at *ICML 2023*. https://arxiv.org/abs/2212.04356

Baevski, Alexei, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations." *NeurIPS 2020*. arXiv:2006.11477. https://arxiv.org/abs/2006.11477

Povey, Daniel, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. "Purely Sequence-Trained Neural Networks for ASR Based on Lattice-Free MMI." *Interspeech 2016*, 2751–2755. https://www.isca-archive.org/interspeech_2016/povey16_interspeech.html

Sekoyan, Monica, Nithin Rao Koluguri, Nune Tadevosyan, Piotr Żelasko, Travis Bartley, Nikolay Karpov, Jagadeesh Balam, and Boris Ginsburg. "Canary-1B-v2 & Parakeet-TDT-0.6B-v3: Efficient and High-Performance Models for Multilingual ASR and AST." arXiv:2509.14128, 17 September 2025. https://arxiv.org/abs/2509.14128

The Whisper model

- An **encoder–decoder Transformer** over 30-second windows of log-Mel spectrogram
- Trained on **680,000 hours** of weakly supervised, multilingual, multitask audio
- It is a **language model conditioned on audio** — and that single fact explains its failure modes
- Multitask by design: transcription, translation, language ID, and timestamps in one model

NOTE: Because Whisper *generates* text rather than *classifying* frames, it can produce fluent output for audio that contains no speech at all.

Radford, Alec, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. "Robust Speech Recognition via Large-Scale Weak Supervision." arXiv:2212.04356, 6 December 2022; published at *ICML 2023*. https://arxiv.org/abs/2212.04356

Many possible ways to run Whisper

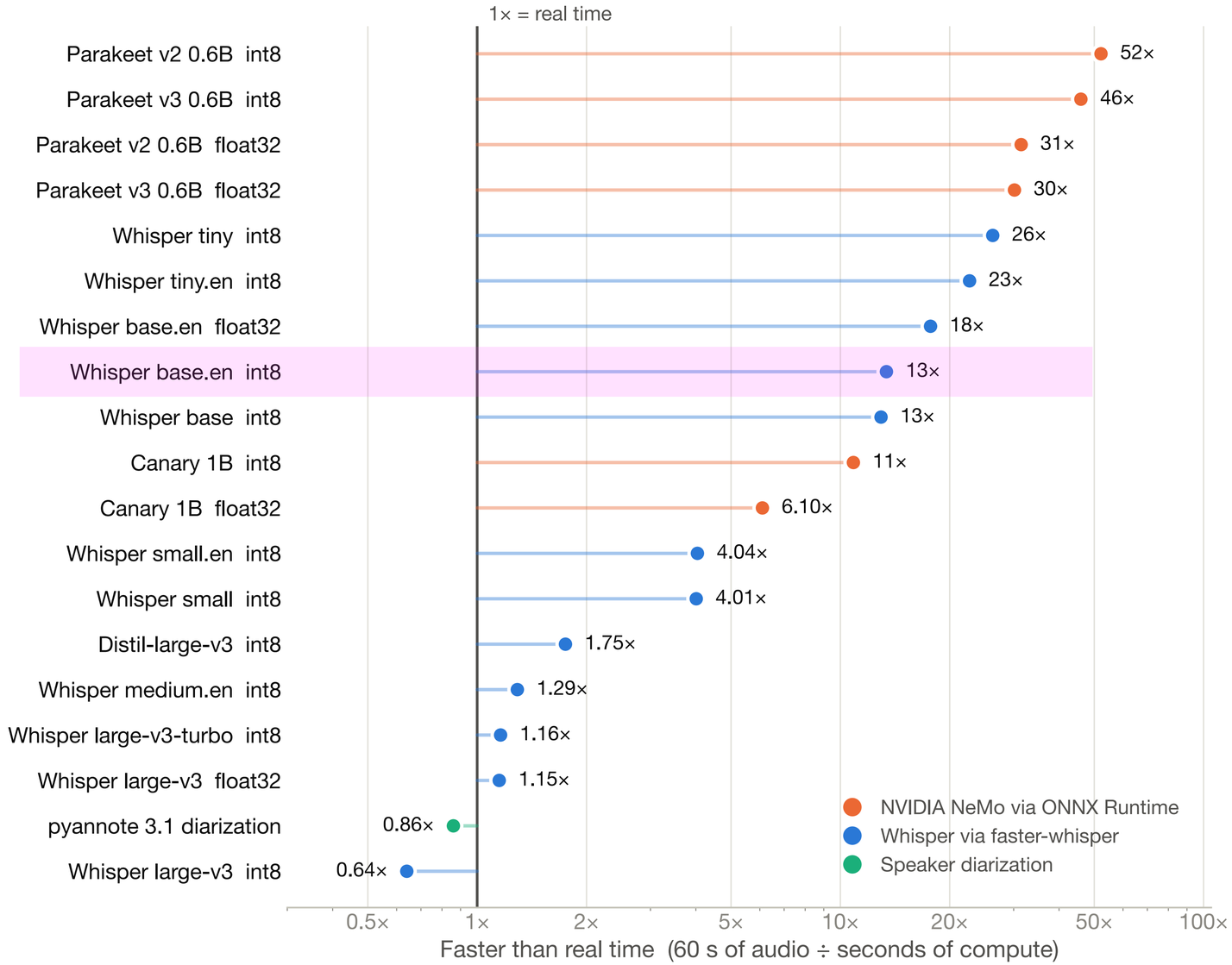
Runtime	Backend	Best for
`openai-whisper`	PyTorch	reference; slowest
`faster-whisper`	CTranslate2	the CPU default (what we use today)
`whisper.cpp`	GGML/Metal	embedded, no Python
`whisperX`	faster-whisper + wav2vec2 + pyannote	word timings + speakers
`MLX-Whisper`, `WhisperKit`	Apple Silicon	Mac-native
`onnx-asr`	ONNX Runtime	runs "GPU-only" NVIDIA models on CPU

- **The weights are the same.** Only the inference engine changes
- 4–5× speedups over the PyTorch implementation are routine for identical output

`openai-whisper` — https://github.com/openai/whisper, `faster-whisper` — https://github.com/SYSTRAN/faster-whisper, `whisper.cpp` — https://github.com/ggml-org/whisper.cpp
`whisperX` — https://github.com/m-bain/whisperX, `mlx-whisper` — https://github.com/ml-explore/mlx-examples/tree/main/whisper, `WhisperKit` — https://github.com/argmaxinc/WhisperKit
`onnx-asr` — https://github.com/istupakov/onnx-asr, with ONNX exports of the NVIDIA NeMo models at https://huggingface.co/istupakov

Actual Timing Estimates

- 60s clip
- Apple M4 Pro, 14 cores, CPU only
- faster-whisper (CTranslate2), beam 5
- NVIDIA NeMo models via onnx-asr, CPU only



Measured 24 July 2026 for this tutorial. Apple M4 Pro, 14 cores, 51.5 GB, macOS 26.5, **CPU only**, single 60.0 s 16 kHz mono clip, beam size 5.

Speed vs. Language Capabilities

Model	Languages	Params	License	On our CPU
Whisper large-v3	99	1.55B	Apache-2.0	1.15× RT
Parakeet-TDT-0.6B-v2	English only	0.6B	CC-BY-4.0	52.2× RT
Parakeet-TDT-0.6B-v3	25 (all European)	0.6B	CC-BY-4.0	45.9× RT
Canary-1B-v2	25 (all European)	0.98B	CC-BY-4.0	10.9× RT

- Multilingual costs almost nothing: v3 covers 25 languages for a ~10% slowdown against v2's English-only
  - but it still cannot touch the other 74 languages Whisper handles.
- Languages supported by Parakeet v3: bg, hr, cs, da, nl, en, et, fi, fr, de, el, hu, it, lv, lt, mt, pl, pt, ro, sk, sl, es, sv, ru, uk
  - No Mandarin, Arabic, Hindi, Japanese, or any African languages.
- Canary also can do speech translation (En↔24) and emits word-level timestamps

Model cards read 25 July 2026: nvidia/parakeet-tdt-0.6b-v3, nvidia/canary-1b-v2, openai/whisper-large-v3.

CPU vs GPU: What actually needs a GPU?

Model family	Native requirement	CPU-viable?
Whisper (faster-whisper, whisper.cpp)	none	Yes
Distil-Whisper	none	Yes
NeMo Parakeet / Canary (native)	CUDA	No
Parakeet / Canary via ONNX export	none	Yes
pyannote 3.1 diarization	none	Yes — but 0.86x real time (bottleneck!)
wav2vec2 forced alignment	none	Yes

- A GPU buys throughput but not better results
- For a corpus of thousands of hours, GPUs will help some parts; for a study of hundreds of episodes, not worth the setup

Nothing in today's tutorial requires a GPU

DIY vs. Commercial Options

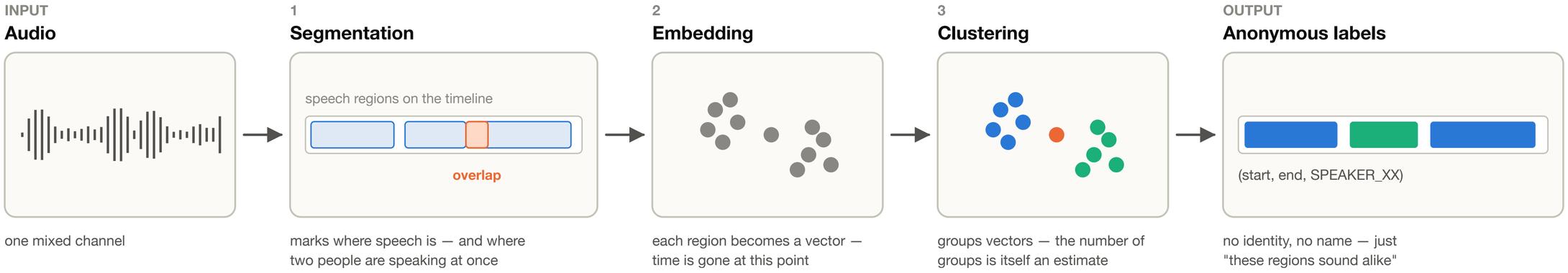
Commercial APIs, per hour of audio:

Service	~Price/hour
OpenAI Whisper API	~$0.36
AssemblyAI	~$0.12–0.37
Deepgram Nova	~$0.26
Self-hosted, CPU	electricity

- A worked example — SPoRC's ~650,000 hours of audio:
  - At $0.22–0.36/hour → $145,000 – $235,000
  - At Parakeet's measured 52× real time → ~12,500 CPU-hours → a few hundred dollars of cloud CPUs, or a few weeks on hardware you already own

Diarization: who spoke when

- pyannote 3.1 is three models in a trench coat:
  - Segmentation: a neural model marks speech regions and local overlap
  - Embedding: each region → a fixed speaker vector
  - Clustering: agglomerative clustering groups vectors into speakers
- Output: (start, end, SPEAKER_XX) — anonymous, arbitrary, and not stable across files
- Metric: DER (Diarization Error Rate) = missed + false alarm + confusion



Diarization is the slowest stage in the whole pipeline

Bredin, Hervé. "pyannote.audio 2.1 speaker diarization pipeline: principle, benchmark, and recipe." *INTERSPEECH 2023*. https://www.isca-archive.org/interspeech_2023/bredin23_interspeech.html

Plaquet, Alexis, and Hervé Bredin. "Powerset multi-class cross entropy loss for neural speaker diarization." *INTERSPEECH 2023*. https://www.isca-archive.org/interspeech_2023/plaquet23_interspeech.html

Model card, `pyannote/speaker-diarization-3.1` — https://huggingface.co/pyannote/speaker-diarization-3.1.

Speaker identification

Problem	The question	Where the answer comes from
Role	host or guest?	speaking time, turn structure, RSS metadata
Naming	which person is this?	the transcript ("I'm your host…"), the episode description
Linking	the same person as in episode 47?	matching names, or voices, across files

- Diarization stops at SPEAKER_00 and SPEAKER_01.
- Speaker recognition is generally from text and/or metadata
- SPoRC 1.0's procedure:
  - spaCy NER over the description and first 350 words
  - Trained RoBERTa model labels each name HOST / GUEST / NEITHER (0.88 accuracy)
  - the host is heuristically mapped to "the first voice which says the host's name."
- SPoRC 2.0 (*TBR*) uses LLMs with metadata only for ~0.9 precision at 0.75 recall.

Forced alignment: word-level time

- Diarization and ASR give segment times but not the start/end times for words
  - You may want these for sociolinguistic / pronunciation analysis
- Forced alignment fits a transcript to the audio and returns a timestamp per word
- **whisperX** — wav2vec2 DTW alignment, post generation
- **Parakeet** — build in alignment during ASR steps (cheap)
- **MFA** — Kaldi-based, needs a pronunciation dictionary, gives phone-level output

Bain, Max, Jaesung Huh, Tengda Han, and Andrew Zisserman. "WhisperX: Time-Accurate Speech Transcription of Long-Form Audio." *INTERSPEECH 2023*. arXiv:2303.00747. https://arxiv.org/abs/2303.00747

McAuliffe, Michael, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. "Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi." *Interspeech 2017*, 498–502. https://doi.org/10.21437/Interspeech.2017-1386, https://montreal-forced-aligner.readthedocs.io/

Prosody: "I didn't say I told her"

- openSMILE / eGeMAPS — the standard 88-parameter acoustic feature set
- F0 (pitch), formants, loudness, jitter/shimmer, spectral slope, MFCCs
- Derived measures researchers actually use: speaking rate, pause structure, turn latency, overlap duration, pitch range
- Two warnings that decide whether this is analysis or artifact:
  - Normalize per speaker. Raw F0 mostly measures vocal-tract length. Lobanov (z-score within speaker) is the standard fix
  - Recording conditions are confounds. A studio mic and a phone line differ acoustically far more than two people do

Prosody gives you an extra channel of information to analyze

Eyben, Florian, Klaus R. Scherer, Björn W. Schuller, Johan Sundberg, Elisabeth André, Carlos Busso, et al. "The Geneva Minimalistic Acoustic Parameter Set (GeMAPS) for Voice Research and Affective Computing." *IEEE Transactions on Affective Computing* 7(2), 2016, 190–202. DOI 10.1109/TAFFC.2015.2457417. https://sail.usc.edu/publications/files/eyben-preprinttaffc-2015.pdf

openSMILE — https://github.com/audeering/opensmile. eGeMAPSv2 i

Lobanov, Boris M. "Classification of Russian Vowels Spoken by Different Speakers." *Journal of the Acoustical Society of America* 49(2B), 1971, 606–608. DOI 10.1121/1.1912396

You probably want a queue infrastructure for scale

- **Idempotency** — key work by episode GUID; re-running must be free
- **Persistent models** — avoid model-loading overhead by having models persist in memory. Move work to them
- **Isolate the unit of failure** — one episode per worker process, so a crash costs one episode
- **Persist immediately** — write each transcript as it completes
- **Rate-limit politely** — you are a guest on someone's CDN
- **Dead-feed queue** — failures are *data* about the ecosystem

Validation steps before you model

- **Hand-transcribe ~10 random minutes.** Compute Word Error Rate (WER) on your audio
  - You *could* start from an ASR transcription but be aware of bias
- **Stratify it** — WER per speaker group you intend to compare
  - If you have different speaker groups (age, gender, dialect, etc.)
- **Check the diarization join** — do speaker turns line up with the words at boundaries?
- **Look at the tails** — the longest and shortest turns are usually where the pipeline breaks
- **Count for repeated n-grams** — a classic Whisper failure, trivially detectable
- **Listen to five random segments with no detected speech** — hallucination check

If you can quote a WER for your own corpus, you're ahead of most published podcast work.

The pipeline, in a nutshell

Challenge	Status
Transcription	Solved enough
Diarization	Works, but is the compute bottleneck
Speaker ID	The weak link.
Alignment	Cheap and usually worth it
Prosody	Available, but normalize or you'll measure microphones

Let’s look at some of the code