Mistral Voxtral TTS: Open-Weight Voice AI You Can Run Locally

March 30, 2026 · by Mark Bartlett

Download this guide as PDF

Quick Answer: Voxtral TTS is Mistral's 4B parameter open-weight text-to-speech model, released March 26, 2026. In blind human tests, 62.8% of listeners preferred it over ElevenLabs Flash v2.5. It clones voices from 3 seconds of audio, supports 9 languages, and hits 70ms time-to-first-audio on an H200. You need 16GB VRAM to run the full model on GPU, or about 2.5GB with the MLX 4-bit version on Apple Silicon (which runs faster than real-time). License is CC BY-NC 4.0, so commercial use requires Mistral's API. For hobbyists and personal projects, it's free and local.

📚 **More on this topic:** Voice Chat with Local LLMs · Crane + Qwen3-TTS Voice Cloning · VRAM Requirements · Building a Local AI Assistant

ElevenLabs charges $22/month for voice cloning and $0.30 per thousand characters on their starter plan. Mistral just gave away something that beats it in blind listening tests.

Voxtral TTS dropped on March 26, 2026, with open weights on HuggingFace. 62.8% of human listeners preferred it over ElevenLabs Flash v2.5 in blind evaluations. It clones voices from 3 seconds of reference audio, speaks 9 languages, and runs on your hardware. No API calls, no subscription, no audio leaving your machine.

There are catches. The license restricts commercial use. The GPU requirements are higher than you'd expect from a "3B" model. And 9 languages is a lot fewer than ElevenLabs' 32. But for personal use and hobbyist projects, this is the best open-weight TTS model available right now.

What Voxtral TTS actually is

Voxtral TTS is three models in a trench coat:

- A 3.4B parameter transformer decoder (built on Ministral 3B) that converts text into semantic tokens
- A 390M parameter flow-matching acoustic transformer that turns semantic tokens into audio latents

- A 300M parameter neural audio codec that decodes those latents into 24kHz audio

Total: about 4.1B parameters. Mistral rounds to “4B” on HuggingFace. Some articles say “3B” because that’s the backbone size, but you’re running all three components.

The model generates audio autoregressively, streaming tokens as it goes. The semantic stage produces tokens at 12.5Hz (one token per 80ms of audio), then the flow-matching stage runs 16 function evaluations to produce acoustic latents. The codec decodes those to waveform.

The result: 70ms time-to-first-audio on an NVIDIA H200 with a 10-second voice reference and 500 characters of input. That’s fast enough for real-time conversation.

How it sounds

The headline number: 62.8% of blind listeners preferred Voxtral over ElevenLabs Flash v2.5 on flagship voices. For voice customization tasks (adapting to a new speaker from a short clip), that number jumped to 69.9%.

Against ElevenLabs v3, their premium tier, Voxtral reaches parity on naturalness and emotional expressiveness. It doesn’t beat v3 outright, but matching a paid premium product with open weights is something.

Voice cloning works from as little as 3 seconds of reference audio, though Mistral recommends 5-25 seconds for best results. The model picks up accent and intonation, and it reproduces disfluencies like “uh” and “um” rather than smoothing them out. Cross-lingual cloning works too: give it a French speaker’s voice and ask it to speak English, and you get English with a French accent.

20 preset voices ship with the model, covering all 9 supported languages: English, French, German, Spanish, Dutch, Portuguese, Italian, Hindi, and Arabic.

Hardware requirements

This is where you should read carefully, because the marketing (“runs on a smartphone”) and the reality are different.

Setup	VRAM/RAM	Speed	Notes
		70ms TTFA, RTF 0.103	

Setup	VRAM/RAM	Speed	Notes
NVIDIA GPU (full BF16)	16GB+ VRAM		H200 benchmarked. A100/4090 should work
MLX 4-bit (Apple Silicon)	~2.5GB RAM	RTF 0.97 (short), 0.74 (long)	Faster than real-time on M-series
MLX 6-bit (Apple Silicon)	~3.5GB RAM	RTF 1.15 (short)	Slightly slower than real-time
MLX BF16 (Apple Silicon)	~8GB RAM	RTF 6.5	Too slow for real-time

RTF is real-time factor. Under 1.0 means the model generates audio faster than you can listen to it. The MLX 4-bit version at RTF 0.97 is right at the boundary of real-time on short clips and comfortably faster on longer ones.

The 16GB VRAM floor for GPU inference means an RTX 4060 Ti 16GB or RTX 3090 at minimum. An RTX 4080 or 4090 would be more comfortable. The model hasn't been benchmarked on consumer NVIDIA GPUs yet, so exact tok/s numbers for a 4090 are still pending.

For Apple Silicon users, the MLX 4-bit version is the sweet spot. 2.5GB leaves plenty of room for an LLM alongside it. An M1 Pro or later with 16GB should handle Voxtral plus a 7B language model simultaneously.

How to run it locally

GPU setup (vLLM)

vLLM added Day-0 support. You need version 0.18.0 or later plus the vLLM-Omni extension:

```
# Install vLLM
pip install -U vllm

# Install the omni extension for audio support
pip install git+https://github.com/vllm-project/vllm-omni.git --upgrade

# Serve the model
vllm serve mistralai/Voxtral-4B-TTS-2603 --omni
```

Then generate audio from Python:

```
import io
import httpx
import soundfile as sf

payload = {
    "input": "The local AI voice stack is finally here.",
    "model": "mistralai/Voxtral-4B-TTS-2603",
    "response_format": "wav",
    "voice": "casual_male",
}

response = httpx.post(
    "http://localhost:8000/v1/audio/speech",
    json=payload,
    timeout=120.0,
)
audio, sr = sf.read(io.BytesIO(response.content), dtype="float32")
```

The server exposes an OpenAI-compatible `/v1/audio/speech` endpoint, so anything that talks to OpenAI's TTS API can point at your local instance instead.

Apple Silicon setup (MLX)

Two commands:

```
pip install -U mlx-audio

python -c "
from mlx_audio.tts.utils import load
model = load('mlx-community/Voxtral-4B-TTS-2603-mlx-4bit')
for result in model.generate(text='Hello from Voxtral.', voice='casual_male'):
    print(f'Generated {result.audio_duration}s of audio')
"
```

The MLX 4-bit model downloads about 2.5GB. First generation takes a minute while it compiles the compute graph, then subsequent runs are fast.

Docker

vLLM publishes a Docker image with everything pre-configured:

```
docker run --gpus all -p 8000:8000 \
    vllm/vllm-openai:v0.18.0 \
    --model mistralai/Voxtral-4B-TTS-2603 \
    --omni
```

How it compares to other open TTS models

Model	Params	Languages	Voice Cloning	Min Reference	VRAM	License	Real-time?
Voxtral TTS	4B	9	Yes	3 sec	16GB GPU / 2.5GB MLX	CC BY-NC 4.0	Yes
Qwen3-TTS	1.7B	10	Yes	3 sec	~4GB	Apache 2.0	Yes (RTF 0.87)
Sesame CSM	1B	English	No	N/A	~4GB	Apache 2.0	Yes
Coqui XTTS v2	~1.6B	17	Yes	6 sec	8GB	MPL 2.0	Yes
Bark	~1.5B	13+	No	N/A	12GB	MIT	No
Piper	Tiny	30+	No	N/A	CPU only	MIT	Yes

Voxtral wins on voice quality in blind tests. But notice a few things:

Qwen3-TTS is less than half the size, runs on 4GB VRAM, has an Apache 2.0 license (commercial use allowed), and supports 10 languages. It scored 0.789 speaker similarity vs ElevenLabs' 0.646 in benchmarks. If you need commercial rights or have limited VRAM, Qwen3-TTS is the better pick.

Coqui XTTS v2 supports 17 languages including Chinese, Japanese, Korean, Russian, and Turkish. If you need CJK or a language Voxtral doesn't cover, XTTS is your only real option among open models. The project's company shut down in 2024, but the community maintains the code.

Piper runs on a Raspberry Pi. It doesn't clone voices and the quality is lower, but it works on hardware where nothing else will.

Sesame CSM is specialized for multi-speaker conversation (two speakers taking turns). English only, but good at what it does.

The license issue

Voxtral's weights are CC BY-NC 4.0. The "NC" means non-commercial. You can download it, run it, modify it, and share it for personal and research use. You cannot ship a product with it or use it in a business workflow without licensing it through Mistral's API ($0.016 per thousand characters).

This is a real limitation. Qwen3-TTS (Apache 2.0) and Piper (MIT) have fully permissive licenses. If you're building something commercial, Voxtral is free to evaluate but not free to deploy.

For hobbyists running a personal voice assistant or reading articles aloud while cooking, this doesn't matter at all.

Building a full local voice pipeline

The killer setup right now: Whisper (STT) to a local LLM to Voxtral (TTS). All offline, all on your hardware.

The latency budget for a voice conversation that feels natural is about 1-2 seconds total. Here's where the time goes:

Stage	Model	Time	VRAM
Speech-to-text	faster-whisper (turbo)	~200ms	~1GB
Language model	Qwen 3.5 9B (Q4)	~500ms to first token	~6GB
Text-to-speech	Voxtral MLX 4-bit	~90ms to first audio	~2.5GB
Total		~800ms	~9.5GB

Under a second. On a MacBook Pro with 16GB of unified memory, that whole stack fits with room to spare. On a desktop with an RTX 3090 (24GB), you could run all three plus a larger language model.

Open WebUI already supports custom TTS backends through its API integration. Point it at your vLLM Voxtral server and you get voice chat in a browser.

What to actually do

If you have Apple Silicon with 16GB+: install the MLX 4-bit version. It's the lowest-friction path and runs faster than real-time. You'll be generating speech in under 5 minutes.

If you have an NVIDIA GPU with 16GB+ VRAM: use vLLM. The Docker image is the cleanest setup. The OpenAI-compatible endpoint means you can swap Voxtral into any existing pipeline that uses OpenAI's TTS.

If you need commercial rights: use Qwen3-TTS instead. Smaller, permissively licensed, and the quality gap with Voxtral is narrow.

If you need more than 9 languages: Coqui XTTS v2 covers 17, Piper covers 30+. Voxtral may add more languages later, but right now CJK and Slavic languages aren't supported.

A year ago, building a local voice assistant meant stitching together half-broken projects and tolerating robotic output. Whisper plus a local LLM plus Voxtral gets you a voice pipeline that never phones home and sounds like a paid service. That's new.

Related guides

- Voice Chat with Local LLMs: Whisper + TTS Setup
- Crane + Qwen3-TTS: Local Voice Cloning with Rust
- How Much VRAM Do You Need for Local LLMs?
- Building a Local AI Assistant

Get notified when we publish new guides.

Subscribe — free, no spam

Source: https://insiderllm.com/guides/mistral-voxtral-tts-local-voice-ai/

Free guides for running AI locally