How to "Mod" Your Own Speech Recognition Engine

MOD9 TECHNOLOGIES

Have you ever wanted to operate your own automatic speech recognition (ASR) Engine?

The benefits are compelling, since you can:

- guarantee data privacy and service availability;
- optimize for speed, accuracy or latency;
- enable use cases that are otherwise cost-prohibitive.

This technical challenge might seem too daunting at first, but let's try to break it down as a series of "mods" you can follow.

Mod 1: Choose a Baseline

As a starting point, our ASR Engine should be compatible with one of the major cloud-hosted platforms. This will let you easily swap in another service if it's ever necessary to provide extended coverage for any of their 100+ languages that may be difficult to support. Google Cloud Speech-to-Text is a good choice, thanks to its well-documented client libraries.

Mod 2: Python SDK and REST API

Google provides a Python library and code samples so that developers can get started with real-time speech recognition; they also provide a REST endpoint to facilitate processing of pre-recorded audio. Don't reinvent the wheel: let's make an SDK and API that precisely adhere to these well-designed specs. This way, porting your code will only require single-line modifications to drop-in our replacements.

Mod 3: Support Additional Clients

While we're at it, let's add a WebSocket interface so web browsers can send microphone input to your Engine. For easier command-line debugging, we can expose a simple JSON messaging protocol over TCP. We can also provide a C++ library in case you might need to optimize low-level integration in streaming media servers.

Mod 4: Build for Linux and Deploy with Docker

You'll want this Engine to be a statically linked binary so it will run on any Linux system, without installing dependencies. We can also make it easy to deploy REST and WebSocket services over encrypted HTTPS by packaging these Engine wrappers in a Docker image.

Mod 5: Calculate Your Costs

Your Engine should be able to process a pre-recorded file 5x faster than its duration using just one CPU. (Equivalently: each CPU should handle up to 5 real-time audio streams at once.) So you could process about 500 hours of audio in one hour of compute time if you had a server with 96 CPUs. Such an *ec2.metal* instance can be rented for just a few dollars an hour. At scale, then, the cost to run your own Engine can be mere hundredths of a cent per minute.

Mod 6: Train Some Models, Repackage Others

Training ASR models is difficult, requiring PhD-level expertise with research-grade software such as Kaldi. Large quantities of high-quality data might only be obtained for a couple of major languages, like English and Spanish. Fortunately, since we've chosen to follow a very popular open-source project, you'll be able to leverage dozens of freely licensed models that have already been trained and generously contributed by the community.

Mod 7: Be Ready To Adapt

Even with excellent pre-trained models, no ASR system will be perfectly tuned for unexpected use cases with domain-specific jargon. Let's make sure that your Engine can be customized on-the-fly, allowing new words and pronunciations to be added or corrected — even in the midst of recognizing an audio stream. You might also want to load your own grammars on-demand for directed dialog flow scenarios, as well as boosting phrases that should be recognized more often (or negatively bias them so they don't re-appear as errors).

Mod 8: The Kitchen Sink

This Engine has become fairly advanced by now, so we might as well make everything configurable. Let's provide fine-grained control over memory limits and request timeouts. You may also want to explore innovations like phrase-level alternatives, designed for Elasticsearch indexing to improve audio search recall (yet another "ASR"). Finally, let's attach a natural language processing (NLP) pipeline for capitalization, punctuation, and number formatting — you'll need that post-processing if you want transcripts to be readable.

Mod 9: Don't Do It All Yourself!

Of course, all of these steps have already been implemented in the Mod9 ASR Engine, so you shouldn't have to build this on your own! Check out mod9.io to see how it all works.

To get started with a free evaluation of a publicly accessible Docker image, try this command: **docker pull mod9/asr** ■

Contact **help@mod9.com** if you have any questions.
Or call (HUH) ASK-ARLO to speak with A Real Live Operator.