CEO

Anthony Minessale

I've been working in the telecommunications industry for 20+ years, and my life's work has revolved around transforming telecom to software and making it programmable.

I have been working on making voice bots that don't suck since they sucked. But I'd like to think we made them suck a bit less and made them more accessible to the world. Before FreeSWITCH, IVRs and voicemail used to cost 100k for a physical server that was converted to pure software.

Like the Internet, when it was new to the general public, as soon as LLMs that could be consumed at scale over a reliable API appeared, all of its use cases were almost immediately obvious. I think anyone can conclude that with an LLM generating human language, you can wire it up to text-to-speech and speech-to-text or to an existing chat interface.

My thesis on approaching this technology started with the hardest thing, voice, as making a chat-based interface is almost trivial in comparison. I have a few key tenets that I worked with initially.

1) Focus on the leading technology (OpenAI)

Like with computers, the Internet, and other inflection point changes, you can presume that the core technology will eventually reach equilibrium and that business cases for using the technology will be the main differentiator. So, rather than an early focus on the model, I wanted to focus on the use case (allowing developers to build digital employees). OpenAI has several things about how its APIS work that give a low-level developer more power to exploit and simplify for end-user developers.

2) Leverage the AI in a way that limits its ability to be exposed to sensitive data by combining it with existing voice menu technology

Our stack is designed as an extension of a more massive stack for general scalable telecom services. The AI stack can actually call into a traditional voice menu to collect sensitive info from the user. Then, it can save a meta-data token and just tell the AI that the data was collected without it ever actually interacting with it.

3) Create an environment for developers that is as easy to use as the Web

Our stack is inspired by UNIX and the Web and is, in essence, an extension of those concepts. The AI Agent you create could be considered a new way to look at web forms in a way that allows you to talk to them. Its prompts tell it how to talk to the user and collect info, then use functions to post that data to your servers, where you can feed information back and modify its behavior in several ways. Because our AI stack is one method in an even larger telephony stack, it has all the powers of its parent platform to work with to transfer calls, send SMS, and scale to any number of concurrent calls that can be over phone lines, SIP (VOIP), WebRTC and UCaaS applications on our platform.

4) Be the lowest latency period

Since the first experiment, I have been working on designs to eliminate latency. I have developed several techniques that are solved mostly by ingenuity and deep knowledge of C, and I have one of the most powerful communications platforms on the planet under my belt.

I know the new GPT-4o is trying to short-circuit the whole thing by doing some of the work in a mobile phone and has models that can ingest audio, but they have a billion dollars to burn, so I would expect nothing less. Meanwhile, I think we have a system that actually does useful things and responds at a speed where you might even consider slowing it down in some cases (which is configurable). We are able to use almost any modern TTS natively and minimize latency at all points of orchestration.

Built to scale

Our AI orchestrator is just another cog in a proven powerful engine so applications can scale forever with the horizontal implementation of its parent platform. Conversational experience will always be a challenge (it still is with real humans, after all) but I think we offer such a wide array of params and ways to approach it that we will evolve further and further.

Vertically focused

We have an elaborate system called SignalWire AI Gateway (SWAIG) that is built on top of the Agent framework and allows deep integration points that are the building blocks to implement drop-in skills. We have several demos for things like booking a restaurant, ordering flowers, cable technician, MFA bot, and more. Rather than focusing on training models, we strive to leverage the least training possible to build working applications quickly. We have a system in the works that will allow live vectorization of arbitrary documents to provide on-demand

I think most people fake their results and we have a strict rule against that.

Realistic in Scope

This is key. The expectations that are often set by how cool AI is. Sometimes, it causes people to make the wrong assumptions. This is nothing new as we, as engineers, often refer to the National Park vs Bird xkcd comic.

We believe that systems using digital employees created with our stack can do a lot of things immediately and even more as the technology progresses:

- Sorting calls for humans.
- Gathering information ahead vs. keeping customers on hold.
- Being voicemail without being voicemail.
- Acting as personal assistants.

What's different from others?

I've seen a few of the other AI stacks, and while we have some things in common, I think our approach has some key differentiated elements.

Part of a fully Programmable Unified Communications stack.
The AI agent stack is tethered to a horizontally scaling UCaaS, CCaaS, CPaaS platform. You can access the agents from SIP, PSTN, WebRTC UCaaS applications not limiting the experience to over the phone.

Scaling Complexity

As your needs evolve, you can deliver dynamic content to generate on-demand agents or build out SWAIG services to allow your agent to get more and more technically capable.

Context Switching

The agent can be dynamically altered mid-conversation to change its focus or core prompting so you can have a general Agent who suddenly becomes completely focused on finding you a good movie to watch and then suddenly able to book a restaurant reservation. There is also a "steps" mode that will force the agent down a series of pre-determined steps where its core mission is updated as it completes each one. This could be combined with context switching to do a more thorough job.

Sliding Window

Rather than continually posting the entire conversation to the AI, a sliding window can be defined to limit the conversation to a certain number of turns. I have an example where it's wired into an infocom text adventure from the 80's where it gets you to say what you want to do and feeds it to the game (no need to save the whole conversation).

A simple digital employee can be created as a single JSON file that can be hosted on our systems or delivered over webhook.

Multiple Voices, Languages and Fillers

An agent can have a single muti-lingual voice or different voices for each language and can switch between them just by asking it to. Each language can have optional fillers which are a list of phrases to indicate the agent is looking something up where the remote service may contribute enough to latency that it has to say, "ok, one second" or "hold on." For extreme cases like one example where you order flowers and it uses an AI image generator to text you a picture of the flowers you describe, a sound file can be played like pencil scribbling or keyboard typing.

Media Files

The agent can play a looping background sound like people talking in a coffee shop or kids playing in a park or a busy office to add to the experience. Via SWAIG the agent can be instructed to play a file as part of the conversation. For example, I had a "duck debugger" who offers to help you find your problem and pretends to ask a rubber duck who quacks in response to your questions.

Video

If the agent is called over a video enabled line, it can use a series of mp4 files to simulate a state of idle, paying attention, and talking. If it's instructed to play a video file, in place of its avatar, it can stream the video to the remote user. I have plans to eventually have it snapshot your inbound video and "see" what is in the picture.

Advanced Barge Cutoff / Adjustable Latency

To pursue the best conversational experience, the agent defaults to a behavior where if you start talking over each other, not only does it stop and let you win, but it also combines everything you are saying into a single turn and removes any early responses. So if you are talking to the agent with many pauses in your speech pattern and the agent starts answering too soon, just by continuing to talk, it will adjust to the new combined utterances and form a completely new response.

Built-in Post-Prompting and full logging and debugging

A post-prompt can be defined to do a final action such as summarizing the conversation, formulating a pizza order into a json blob, etc. Special SWAIG functions are available at the beginning and end of the conversation to take a leading or final action in particular cases.
Debugging webhooks can be defined to obtain mid-conversation details, so you know what's going on as it works.

Unified support for ASR/TTS

All of our supported ASR/TTS interfaces are available to the AI and the normal voice menu system, so you can mix and match advanced AI portions of the conversation with classic voice menu techniques. Currently, we support Elevenlabs, Google, PlayHT, and Amazon Polly, and adding more is trivial. Any new ones appear in all aspects of the platform.

Persistent Conversation Tracking

Several features enable the agent to be seeded with information from a previous call so you can provide the sense that it knows what happened in previous conversations.

Ability to receive chat or SMS during the conversation

Messages can be sent to the agent while you are talking to it on the phone, allowing you to chat or SMS with it while you are talking to it, for cases like sending your email without reading it out loud or getting helpful links from the agent. The Santa Claus example asks you what you want for Christmas and then sends you links at Amazon. The flowers example asks you what flowers you want and who to send them to and texts them a custom AI-generated picture of flowers based on your input.

SWAIG

The SignalWire AI Gateway is a native interface that is immeasurably flexible as part of our AI stack. It extends the basic 'functions' feature from the language model into a system that allows the developer to create solutions with evolving complexity, starting with static file and template-driven data expansion and moving on to full remote custom webhooks.