The Ultimate Guide to Speech-To-Text Recognition

Speech-to-Text: A quick history

Speech recognition first hit the scene in the 1950s, thanks to Bell Laboratories. The next decade or so saw marginal improvements by various global players, but these early efforts were limited primarily to the recognition of digits or a handful of vowels and consonants.

The pace of progress picked up in the 1970s, thanks to U.S. government interest and investment. The technology was still, however, limited to very specific use cases which sought out specific sound patterns and word templates. IBM, for example, chose to focus on speech-to-text recognition for transcription of business correspondence.

The first real game-changer came in the 1980s, when use of Hidden Markov Model (HMM) statistical prediction expanded speech recognition capabilities from hundreds of words to thousands. Its use removed many of the former limitations by evaluating the likelihood that sounds were words instead of restricting recognition to known words or patterns. This shift to a data-driven methodology allowed for significantly greater accuracy than trying to force computers to process speech like humans.

One major problem remained – the inability to process continuous speech. Users at this point needed to speak slowly, word by word, making speech-to-text recognition more of a novelty than a practical day-to-day tool. It wasn't until 1997 that Dragon introduced the first speech-to-text offering that allowed users to speak normally. With accuracy of about 80%, though, manual entry remained more efficient for most users.

Then Google delivered a major breakthrough in 2008 with its Voice Search, thanks to having the key ingredients that others lacked for improving accuracy – training data from billions of search queries and data centers with massive cloud-computing power. This opened up a world of new possibilities for speech recognition, including the option of voice coding, first demonstrated at PyCon 2013.

Speech-to-Text: Where it stands today

Speech-to-text recognition has become more accessible as more developers gain expertise and experience and more APIs become available to move closer to plug-and-play capability. There are commercial APIs available from the big cloud providers that provide access to their associated power and resources, but there's not much opportunity for customization and access comes at a price. There are also open-source libraries of tools for building a custom system, using a variety of different methods and languages.

There are three main methods used for speech recognition systems. Synchronous Recognition is best for short audio (one minute or less), with the audio data sent and fully processed before results are returned. Asynchronous Recognition works for up to eight hours of audio. The audio data is sent and initiates an ongoing process that can be periodically polled for results. Streaming Recognition is designed for real-time recognition. Audio data is streamed and processed, with results provided as generated.

Python is most commonly used to code speech recognition algorithms and modules. There are also third-party APIs to help save time developing specific capabilities, such as real-time transcription.

Voice as well as speech can be recognized and used to personalize access. Industry-specific terminology, unique vocabulary, a variety of languages, and even accents can be incorporated into systems to optimize accuracy.

While many tools are designed for conversational speech, some are designed specifically for coding. One type allows developers to describe the commands to code without dictating word for word, using an NLP layer to run the speech inputs through machine-learning models that translate them into valid code constructs. Others are more complex and require a more granular understanding of each programming task.

Ongoing improvements in accuracy, speaker diarization, real-time transcription, customer-specific language models, background noise reduction, and related areas are expected going forward.

Speech-to-Text: In action

Speech-to-text recognition is highly relevant for developers today, both within specified capabilities and for development itself.

Feature implementation: Developers are often asked to incorporate various features for use internally or as part of a marketable product. Speech-to-text recognition offers a way to add real-time speech recognition and/or subtitles, transcribe Zoom recordings, query/paginate historical transcripts, improve transcript accuracy, detect and/or redact sensitive content, properly diarize speakers in a conversation, and much more.

Streamlining/Standardizing: Voice-coding platforms allow developers to simply speak to author, edit, manipulate, and navigate code. They can create custom voice commands that accommodate and automate their own specific workflows or that translate to frequently-used code strings or comments for shared use. Greater coding consistency throughout an organization is likely to result in fewer typo-related bugs and streamlined testing/analysis. Offering prepackaged voice “code snippets” may also allow for smoother integration with other systems/APIs.

Collaboration: Speech-to-text translation could allow programmers from different localities to speak in their native language to code. The system would translate and add the code in the core language, allowing for easier collaboration across cultures/countries and minimizing errors associated with coding in a secondary language. A speech interface could also be handy for collaborating on code during a team meeting or class.

User-centered design: People communicate differently in speech than in writing, and the goal is to make it easier for people to get the right result by simply stating what they want. Voice coding creates a deeper familiarity with the nuances of speech and system response. What’s learned during coding informs the approach for other scenarios and encourages a more human-centric approach.

Wellness: Voice coding can help minimize repetitive use injuries often suffered by developers and allow those with existing injuries to continue coding.

Better professional equity/accessibility: Voice coding could open up software development careers to people with untrained natural talent and support greater workplace equity for those who are differently-abled. Anyone who can use logic and structure to verbally express what they want to do could use a voice coding system and let machine learning convert their input into valid code.

Bottom line, speech recognition is widely used across many industries today, wherever there’s a need for customer self-service, call analytics, agent assistance, media monitoring, captioning, transcripts, and much more. Contact us to find out how you can put it to work for your organization.