Sample Code

Recognizing speech in live audio

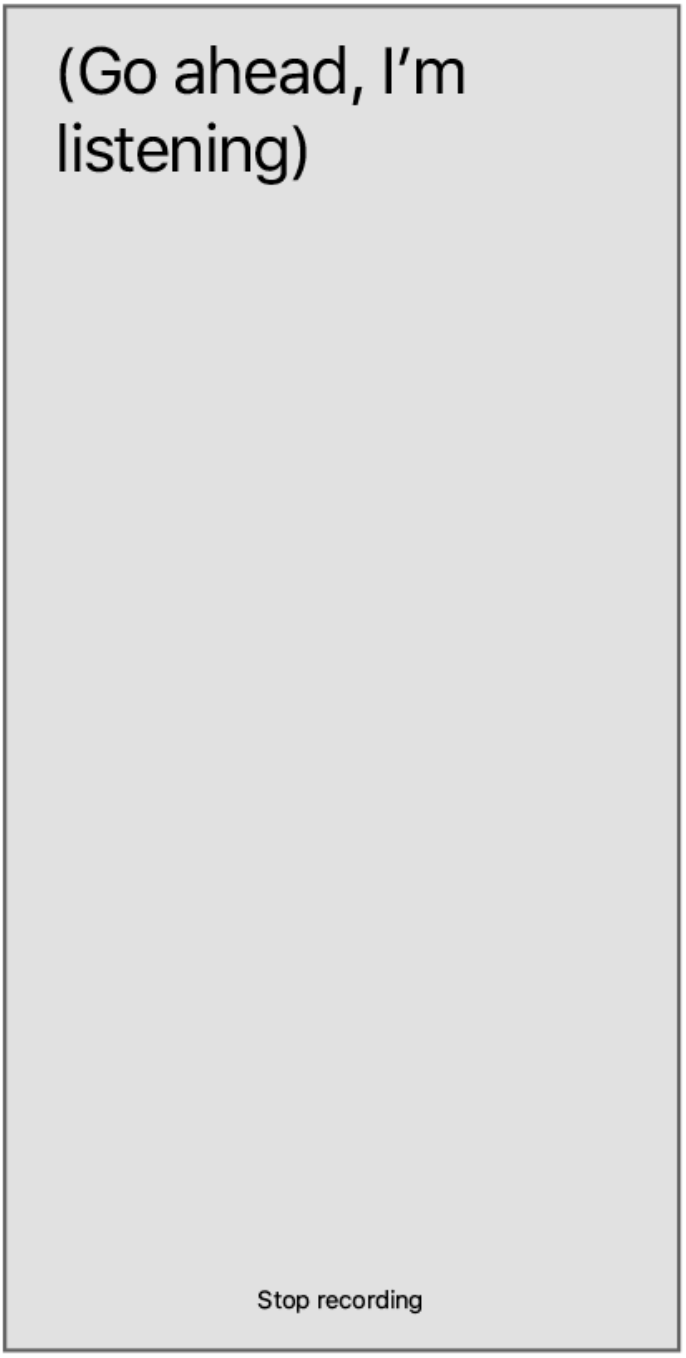
Perform speech recognition on audio coming from the microphone of an iOS device.

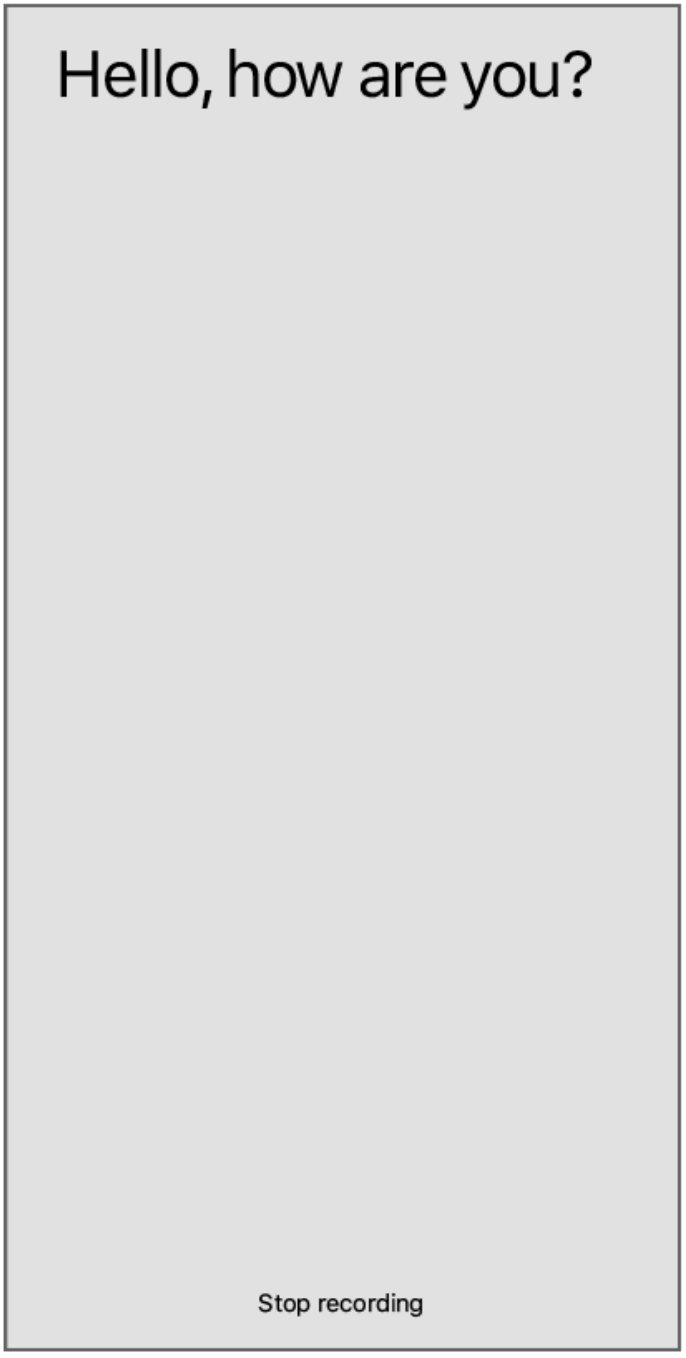
Download

iOS 17.0+ | iPadOS 17.0+ | Xcode 16.0+

Overview

This sample project demonstrates how to use the Speech framework to recognize words from captured audio. When the user taps the Start Recording button, the SpokenWord app begins capturing audio from the device's microphone. It routes that audio to the APIs of the Speech framework, which process the audio and send back any recognized text. The app displays the recognized text in its text view, continuously updating that text until you tap the Stop Recording button. The sample app doesn't run in Simulator, so you need to run it on a physical device with iOS 17 or later, or iPadOS 17 or later.





Important

Apps need to include the `NSSpeechRecognitionUsageDescription` key in their `Info.plist` file and request authorization to perform speech recognition. For information about requesting authorization, see Asking Permission to Use Speech Recognition.

Customize the language model

Developers can enhance the `SFSpeechRecognizer` for specific use cases and applications by customizing its language model. The SpokenWord app uses language model customization to improve accuracy when recognizing certain chess moves. The high-level steps in this process are:

- Training data generation

- Training data preparation
- Request configuration

This sample code project includes a command-line utility named `datagenerator` that produces a training data file. The project includes a sample output file in the SpokenWord app bundle at `customlm/en_US/CustomLMData.bin`. The `datagenerator` utility uses classes that the Speech framework defines to generate data, starting with the SFCustomLanguageModelData, which the `ResultBuilder` DSL builds.

```
let data = SFCustomLanguageModelData(locale: Locale(identifier: "en_US"), identifie
```

Training data samples can include exact phrases that the app is likely to encounter.

```
SFCustomLanguageModelData.PhraseCount(phrase: "Play the Albin counter gambit", count
```

The app can also define phrases using templates, which expand automatically to generate large amounts of data.

```
SFCustomLanguageModelData.PhraseCountsFromTemplates(classes: [
    "piece": ["pawn", "rook", "knight", "bishop", "queen", "king"],
    "royal": ["queen", "king"],
    "rank": Array(1...8).map({ String($0) })
]) {
    SFCustomLanguageModelData.TemplatePhraseCountGenerator.Template(
        "<piece> to <royal> <piece> <rank>",
        count: 10_000
    )
}
```

An app that uses specialized terminology can also define custom vocabulary, complete with pronunciation information.

```
SFCustomLanguageModelData.CustomPronunciation(grapheme: "Winawer", phonemes: ["w I r
SFCustomLanguageModelData.CustomPronunciation(grapheme: "Tartakower", phonemes: ["t

SFCustomLanguageModelData.PhraseCount(phrase: "Play the Winawer variation", count: 1
SFCustomLanguageModelData.PhraseCount(phrase: "Play the Tartakower", count: 10)
```

Configure the microphone using AVFoundation

The SpokenWord app uses AVFoundation to communicate with the device's microphone. Specifically, the app configures the shared `AVAudioSession` object to manage the app's audio interactions with the rest of the system, and it configures an `AVAudioEngine` object to retrieve the microphone input.

```
private let audioEngine = AVAudioEngine()
```

Tapping the app's Start Recording button retrieves the shared `AVAudioSession` object, configures it for recording, and makes it the active session. Activating the session lets the system know that the app needs the microphone resource. If that resource is unavailable — perhaps because the user is talking on the phone — the `setActive(_:options:)` method throws an exception.

```
// Configure the audio session for the app.
let audioSession = AVAudioSession.sharedInstance()
try audioSession.setCategory(.record, mode: .measurement, options: .duckOthers)
try audioSession.setActive(true, options: .notifyOthersOnDeactivation)
let inputNode = audioEngine.inputNode
```

When the session is active, the app retrieves the `AVAudioInputNode` object from its audio engine and stores it in the local `inputNode` variable. The input node represents the current audio input path, which can be the device's built-in microphone or a microphone connected to a set of headphones.

To begin recording, the app installs a tap on the input node and starts up the audio engine, which begins collecting samples into an internal buffer. When a buffer is full, the audio engine calls the provided block. The app's implementation of that block passes the samples directly to the request object's `append(_:)` method, which accumulates the audio samples and delivers them to the speech recognition system.

```
// Configure the microphone input.
let recordingFormat = inputNode.outputFormat(forBus: 0)
inputNode.installTap(onBus: 0, bufferSize: 1024, format: recordingFormat) { (buffer:
    self.recognitionRequest?.append(buffer)
}

audioEngine.prepare()
try audioEngine.start()
```

The SpokenWord app processes the training data that the `datagenerator` utility produces by calling `prepareCustomLanguageModel(for:clientIdentifier:configuration:`

completion:). This method can have a large amount of associated latency, so the system calls it off the main thread.

```
Task.detached {
    do {
        let assetPath = Bundle.main.path(forResource: "CustomLMData", ofType: "bin",
        let assetUrl = URL(fileURLWithPath: assetPath)
        try await SFSpeechLanguageModel.prepareCustomLanguageModel(for: assetUrl,
                                                                   clientIdentifier:
                                                                   configuration: se
    } catch {
        NSLog("Failed to prepare custom LM: \(error.localizedDescription)")
    }
    await MainActor.run { self.recordButton.isEnabled = true }
}
```

Create the speech recognition request

To recognize speech from live audio, SpokenWord creates and configures an SFSpeechAudioBufferRecognitionRequest object. When it receives recognition results, the app updates its text view accordingly. The app sets the request object's shouldReportPartialResults property to true, which causes the speech recognition system to return intermediate results as they are recognized.

```
// Create and configure the speech recognition request.
recognitionRequest = SFSpeechAudioBufferRecognitionRequest()
guard let recognitionRequest = recognitionRequest else { fatalError("Unable to creat
recognitionRequest.shouldReportPartialResults = true
```

The SpokenWord app then configures the request to use language model customization, using the same configuration object that the system provides earlier to prepareCustomLanguageModel. The system needs to service customized requests on the device, which it enforces using the requiresOnDeviceRecognition flag.

```
// Keep speech recognition data on device
if #available(iOS 13, *) {
    recognitionRequest.requiresOnDeviceRecognition = true
    if #available(iOS 17, *) {
        recognitionRequest.customizedLanguageModel = self.lmConfiguration
    }
```

```
}
```

To begin the speech recognition process, the app calls `recognitionTask(with:resultHandler:)` on its `SFSpeechRecognizer` object. That method uses the information in the provided request object to configure the speech recognition system and to begin processing audio asynchronously. Shortly after calling it, the app begins appending audio samples to the request object. When you tap the Stop Recording button, the app stops adding samples and ends the speech recognition process.

Because the request's `shouldReportPartialResults` property is `true`, the `recognitionTask(with:resultHandler:)` method executes its block periodically to deliver partial results. The app uses that block to update its text view with the text in the `bestTranscription` property of the result object. If it receives an error instead of a result, the app stops the recognition process altogether.

```
// Create a recognition task for the speech recognition session.
// Keep a reference to the task so that it can be canceled.
recognitionTask = speechRecognizer.recognitionTask(with: recognitionRequest) { resul
    var isFinal = false

    if let result = result {
        // Update the text view with the results.
        self.textView.text = result.bestTranscription.formattedString
        isFinal = result.isFinal
    }

    if error != nil || isFinal {
        // Stop recognizing speech if there is a problem.
        self.audioEngine.stop()
        inputNode.removeTap(onBus: 0)

        self.recognitionRequest = nil
        self.recognitionTask = nil

        self.recordButton.isEnabled = true
        self.recordButton.setTitle("Start Recording", for: [])
    }
}
```

Respond to availability changes for speech recognition

The availability of speech recognition services can change at any time. For some languages, speech recognition relies on Apple servers, which requires an active Internet connection. If that Internet connection is lost, an app must be ready to handle the disruption of service that can occur.

Whenever the availability of speech recognition services changes, the SFSpeechRecognizer object notifies its delegate. SpokenWord provides a delegate object and implements the speechRecognizer(_:availabilityDidChange:) method to respond to availability changes. When services become unavailable, the method disables the Start Recording button and updates its title. When services become available, the method reenables the button and restores its original title.

```
public func speechRecognizer(_ speechRecognizer: SFSpeechRecognizer, availabilityDic
    if available {
        recordButton.isEnabled = true
        recordButton.setTitle("Start Recording", for: [])
    } else {
        recordButton.isEnabled = false
        recordButton.setTitle("Recognition Not Available", for: .disabled)
    }
}
```

See Also

Audio sources

`class` SFSpeechURLRecognitionRequest

A request to recognize speech in a recorded audio file.

`class` SFSpeechAudioBufferRecognitionRequest

A request to recognize speech from captured audio content, such as audio from the device's microphone.

`class` SFSpeechRecognitionRequest

An abstract class that represents a request to recognize speech from an audio source.