Automatically detect language

cloud.google.com/speech-to-text/docs/multiple-languages

Preview

This feature is covered by the Pre-GA Offerings Terms of the Google Cloud Terms of Service. Pre-GA features might have limited support, and changes to pre-GA features might not be compatible with other pre-GA versions. For more information, see the launch stage descriptions.

This page describes how to provide multiple language codes for audio transcription requests sent to Speech-to-Text.

In some situations, you don't know for certain what language your audio recordings contain. For example, if you publish your service, app, or product in a country with multiple official languages, you can potentially receive audio input from users in a variety of languages. This can make specifying a single language code for transcription requests significantly more difficult.

Multiple language recognition

Speech-to-Text offers a way for you to specify a set of alternative languages that your audio data might contain. When you send an audio transcription request to Speech-to-Text, you can provide a list of additional languages that the audio data might include. If you include a list of languages in your request, Speech-to-Text attempts to transcribe the audio based upon the language that best fits the sample from the alternates you provide. Speech-to-Text then labels the transcription results with the predicted language code.

This feature is ideal for apps that need to transcribe short statements like voice commands or search. You can list up to three alternative languages from among those that Speech-to-Text supports in addition to your primary language (for four languages total).

Even though you can specify alternative languages for your speech transcription request, you must still provide a primary language code in the `languageCode` field. Also, you should constrain the number of languages you request to a bare minimum. The fewer alternative language codes that you request helps Speech-to-Text more successfully select the correct one. Specifying just a single language produces the best results.

Enable language recognition in audio transcription requests

To specify alternative languages in your audio transcription, you must set the `alternativeLanguageCodes` field to a list of language codes in the `RecognitionConfig` parameters for the request. Speech-to-Text supports alternative language codes for all

speech recognition methods: `speech:recognize`, `speech:longrunningrecognize`, and Streaming.

Note: You can only use the alternative languages feature with the `default` or `command_and_search` models. For more information about how to specify different models, see the `RecognitionConfig` reference documentation.

Use a local file

Refer to the `speech:recognize` API endpoint for complete details.

To perform synchronous speech recognition, make a `POST` request and provide the appropriate request body. The following shows an example of a `POST` request using `curl`. The example uses the access token for a service account set up for the project using the Google Cloud Google Cloud CLI. For instructions on installing the gcloud CLI, setting up a project with a service account, and obtaining an access token, see the quickstart.

The following example shows how to request transcription of an audio file that may include speech in English, French, or German.

```
curl -s -H "Content-Type: application/json" \
    -H "Authorization: Bearer $(gcloud auth application-default print-access-
token)" \
    https://speech.googleapis.com/v1p1beta1/speech:recognize \
    --data '{
    "config": {
        "encoding": "LINEAR16",
        "languageCode": "en-US",
        "alternativeLanguageCodes": ["fr-FR", "de-DE"],
        "model": "command_and_search"
    },
    "audio": {
        "uri": "gs://cloud-samples-tests/speech/commercial_mono.wav"
    }
}' > multi-language.txt
```

If the request is successful, the server returns a `200 OK` HTTP status code and the response in JSON format, saved to a file named `multi-language.txt`.

```
{
  "results": [
    {
      "alternatives": [
        {
          "transcript": "hi I'd like to buy a Chromecast I'm ..."
          "confidence": 0.9466864
        }
      ],
      "languageCode": "en-us"
    },
    {
      "alternatives": [
        {
          "transcript": " let's go with the black one",
          "confidence": 0.9829583
        }
      ],
      "languageCode": "en-us"
    },
  ]
}
```


Use a remote file

JavaNode.js
View on GitHub Feedback

```
/**
 * Transcribe a remote audio file with multi-language recognition
 *
 * @param gcsUri the path to the remote audio file
 */
public static void transcribeMultiLanguageGcs(String gcsUri) throws Exception {
  try (SpeechClient speechClient = SpeechClient.create()) {
    ArrayList<String> languageList = new ArrayList<>();
    languageList.add("es-ES");
    languageList.add("en-US");
    // Configure request to enable multiple languages
    RecognitionConfig config =
        RecognitionConfig.newBuilder()
            .setEncoding(AudioEncoding.LINEAR16)
            .setSampleRateHertz(16000)
            .setLanguageCode("ja-JP")
            .addAllAlternativeLanguageCodes(languageList)
            .build();
    // Set the remote path for the audio file
    RecognitionAudio audio = RecognitionAudio.newBuilder().setUri(gcsUri).build();
    // Use non-blocking call for getting file transcription
    OperationFuture<LongRunningRecognizeResponse, LongRunningRecognizeMetadata>
response =
        speechClient.longRunningRecognizeAsync(config, audio);
    while (!response.isDone()) {
      System.out.println("Waiting for response...");
      Thread.sleep(10000);
    }
    for (SpeechRecognitionResult result : response.get().getResultsList()) {
      // There can be several alternative transcripts for a given chunk of speech.
Just use the
      // first (most likely) one here.
      SpeechRecognitionAlternative alternative =
result.getAlternativesList().get(0);
      // Print out the result
      System.out.printf("Transcript : %s\n\n", alternative.getTranscript());
    }
  }
}
```

Was this helpful?