Travis Payton

GOOGLE SPEECH API

INFORMATION AND GUIDELINES

ABSTRACT

Google's Speech recognition system is powerful, but still not readily available for public use. This paper covers how to use the Google Speech API, from acquiring API Keys to the parameters in the requests that are understood so far. Google has made it clear that this API is not for public use, and anybody wishing to use it should seek express consent from Google. Seeing how the web is an open standard, this paper only explores how the API works and in no way condones the misuse of the service.

INTRODUCTION

This paper aims to cover everything a person needs to know in order to access the Speech API using any tool or language they choose. Currently the API is used primarily by Google Chrome's Speech Input Javascript API, which is defined in the W3C Web Speech API Specification [1].

For the purposes of this paper we will only be exploring how Chrome interacts with the speech recognition API, and not on how to use their Javascript extension.

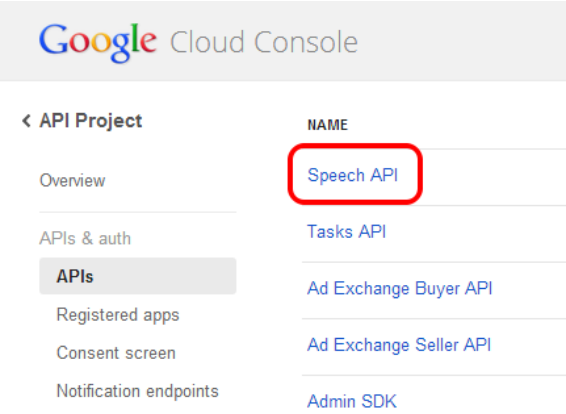
The parts of this paper are as follows:

1. Acquiring an API Key
2. Speech API v1
3. Speech API Full-Duplex
4. Common Parameters

1. ACQUIRING AN API KEY

1.1 DEVELOPER API KEY

In order to gain access to an API key for Google Speech, you will need to be a member of the Chromium Development Group[2], once you are a member of the group, a new service will appear on your list of API's in the



The API Key that you get from being a member of the Chromium Dev group only allows you access to 50 calls a day. Fairly limiting, especially since you will probably make that many in just trying to figure out the API.

1.2 CHROME API KEY

Another option is to use the API Key built in to Chrome. You can do this by capturing the headers that Chrome sends when using the speech input in Chrome. Since Chrome talks to the Speech API via TLS/SSL using a packet capture tool like Wireshark or tcpdump will not be effective. Instead you can use Chrome's

[1] https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html, Glen Shires & Hans Wennborg, Google Inc., 19 October 2012, W3C.

[2] Chromium Development Group: https://groups.google.com/a/chromium.org/forum/#!forum/chromium-dev

built-in header capture debug tool capture the full headers in plain text. This is built into chrome by typing "*about:net-internals*" into the omnibar. You can then export the captured events to a json file and view it with your favorite text editor.

Once you capture yourself using the Speech input of chrome, you can then go through the exported file and find the headers of Chrome calling their server.

Here you can see the full header:

> host":"www.google.com",":
> method":"POST",":path":"/s
> peech-api/full-
> duplex/v1/up?key=AIzaSyBO
> ti4mM-
> 6x9WDnZIjIeyEU21OpBXqW
> Bgw&pair=3B57192471F0F1
> 83&output=pb&lang=en-
> US&pFilter=2&maxAlternati
> ves=1&client=chromium&co
> ntinuous&interim

We can not only see the key that Chrome is using, but also the full path and parameters that they are using. For more about the full-duplex API, please see section 3.

The key Chrome uses is more than likely hard coded into the browser. Every browser tested so far has had the same key supporting that idea. The nice thing about this is that there should be no limits on the number of API requests that can be made with this Key as everyone in the world who uses chrome is using the same key.

2. SPEECH API VERSION 1

The first information that became publicly available concerning Google's speech recognition referenced the one shot API endpoint. https://www.google.com/speech-api/v1/recognize.

This endpoint only allows at most a 40 second clip and limited file size. The exact specifications are not known, but initial testing proves that this endpoint is only good for a few words at best. Mike Pultz has a great article on how to use this API endpoint with curl in PHP[3]. This endpoint is the easiest to use.

Basically this is a POST request to the URL above, with the audio binary in the body of the post. A Content-Type header must be specified in the following format:

```
Content-Type: audio/x-flac; rate=16000
```

Where rate is the sampling rate of the file. If this is incorrect then Google will have a hard-time recognizing the audio.

The API takes 2 known formats: FLAC and Speex. However the Speex implementation does not seem to be the standard format and so the recommended format is FLAC.

When you call this API you will pass the parameters regarding your request in the URL.

For example:

```
…api/v1/recognize?xjerr=1&client=chrom
ium&lang=en-US
```

Tells the server that the file is in English-US, and that the error tolerance should be set to 1, and that the requesting client is chromium. For

[3] http://mikepultz.com/2011/03/accessing-google-speech-api-chrome-11/, Accessing Google Speech API / Chrome 11, Mike Pultz, March 2011.

more information about known parameters see section 4.

A successful call to this API endpoint will return the results in JSON, with an array of possible responses and their Neural Network's confidence that that is correct.

```
{
    "status": 0,
    "id":
"b3447b5d98c5653e0067f35b32c0a8ca-1",
    "hypotheses": [
    {
        "utterance": "i like pickles",
        "confidence": 0.9012539
    },
    {
        "utterance": "i like pickle"
    }]
}
```

3 FULL-DUPLEX API

The non-duplex version of the API is limited to short and small files. So even though it is easy to use and doesn't require an API key to access, the functionality is rather limited. The other API available is the full-duplex API which is what the Speech Input in Chrome uses.

The important thing to remember with this API endpoint is that it is **full-duplex**, meaning that you have to have both on an upstream connected uploading the audio, and a downstream connected downloading the transcription. If the upstream disconnects, so does the downstream

Again, Mike Pultz has a great blog article on how to use this with PHP and cURL[4]. In his original post about the API, he mentions briefly that he was able to reverse engineer it from the Chromium Source code[5]. An excellent resource that is very valuable in understanding how it works. The google_streaming_remote_engine.cc file is the guts of how Chromium interacts with the full duplex API. (For more reference on the non-duplex API see the google_one_shot_remote_engine.cc file).

Instead of just sending one POST request with the file and getting a transcription back in response, you have to send a POST with the file and a GET request simultaneously. See the ConnectBothStreams function.

We will first start with the POST.

3.1 POST

The URL for the POST request is

https://www.google.com/speech-api/full-duplex/v1/up

You need to pass the following parameters:

key = API KEY

pair = a random string used to connect the down stream to this string.

output = the type of response you want, if none is specified, JSON is used.

This stream must be connected first. A GET request to "down" without a matching POST to "up" will fail. Also, if the upstream terminates before the full transcription is finished downloading, then the down stream will be terminated prematurely. So if you are using a tool like cURL you can limit your upload speed so that it doesn't finish too soon.

3.2 GET

The URL for the GET request is:

[4] http://mikepultz.com/2013/07/google-speech-api-full-duplex-php-version/, Mike Pultz, Google Speech API – Full Duplex PHP Version, June 2013

[5] http://src.chromium.org/viewvc/chrome/trunk/src/content/browser/speech/, Chromium Source Code for the Speech API.

https://www.google.com/speech-api/full-duplex/v1/down

The only parameters you need are the key and pair. The rest of the parameters are included in the POST request to “up?”.

You will receive a stream back of the latest transcription results as your up stream is uploading. Once Google is finished processing the upstream, it will pass a final to the downstream. So it is best to keep listening for final on the down stream and only disconnect the upstream once you’ve received it.

Next we will talk about some of the extra parameters that you can send in these requests:

4 PARAMETERS

Some common Parameters which you can pass to the Full-duplex API

Key = the API key for the service.

pFilter = profanity filter, 0 = Off, 1= medium? 2=Strict

lang = The language of the recording / transcription. Use the standard webcodes for your language. I.e. en-US for English-US, ru for Russian, etc.

output = The output of the stream. Some values include “pb” for binary, “json” for json string.

Pair = required for using the full-duplex stream. A random alphanumeric string at least 16 characters long used in both up and down streams.

maxAlternatives = How many possible transcriptions do you want? 1 – X.

continuous = Used in full-duplex to keep the stream open and transcoding as long as there is no silence.

Interim = tells chrome to send back results before its finished, so you get a live stream of possible transcriptions as it processes the audio.

For the one_shot api there are a few other options

lm = Grammars to use – not sure how to use this, believe its used to specify the type of audio, ie transcription, message, etc.

xhw = hardware information – again, not sure how to use it.

CONCLUSION

The Google API is a great resource, and hopefully once Google releases it publicly, the documentation will be more complete and the service easier to implement. For an easy transcription service, check out Nuance’s API.