Detect text in images

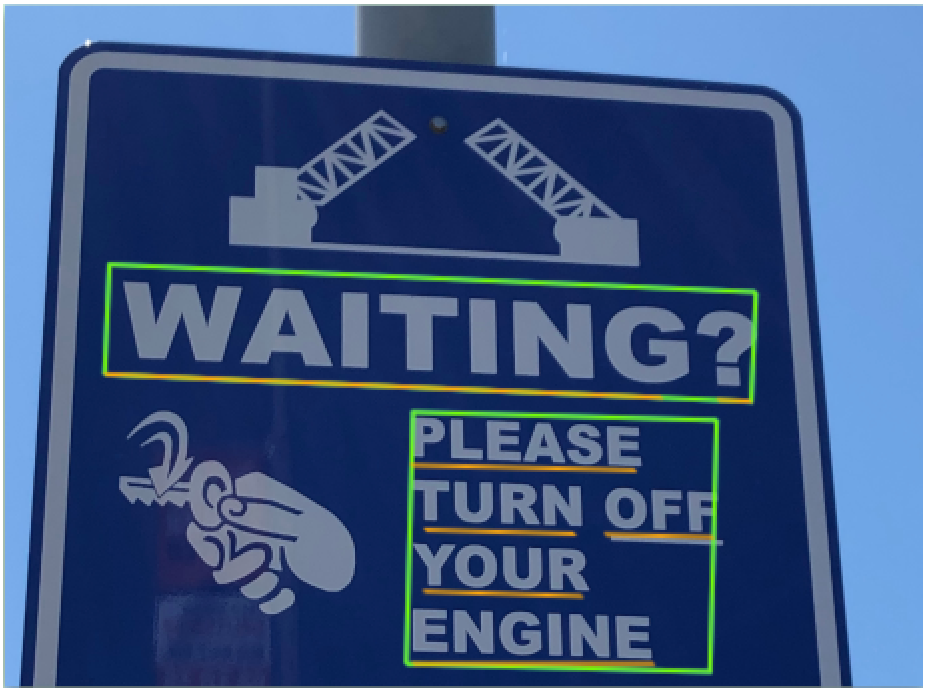
Using this API in a mobile app? Try ML Kit for Firebase (https://firebase.google.com/docs/ml-kit/), which provides native Android and iOS SDKs for using Cloud Vision services, as well as on-device ML Vision APIs and on-device inference using custom ML models.

Optical Character Recognition (OCR)

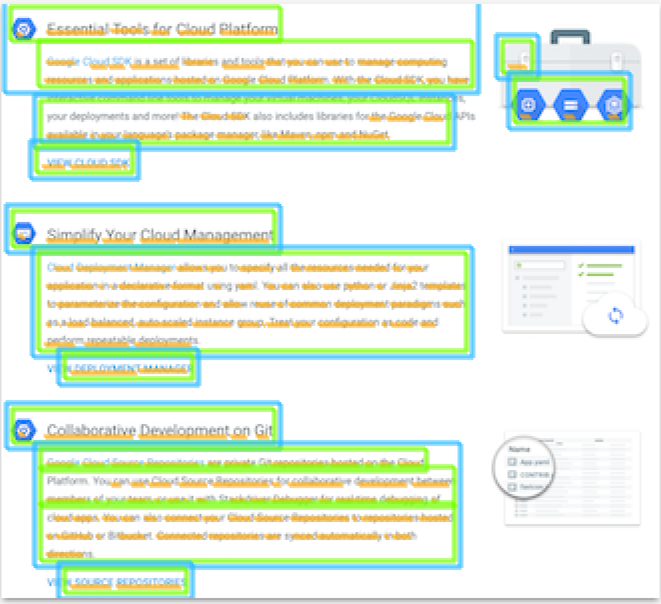
Note: Cloud Vision now supports offline **asynchronous batch image annotation** for all features. This asynchronous request supports up to 2000 image files and returns response JSON files that are stored in your Google Cloud Storage bucket. For more information about this feature, refer to Offline batch image annotation (https://cloud.google.com/vision/docs/batch).

The Vision API can detect and extract text from images. There are ***two*** annotation features that support optical character recognition (OCR):

- `TEXT_DETECTION` detects and **extracts text from any image**. For example, a photograph might contain a street sign or traffic sign. The JSON includes the entire extracted string, as well as individual words, and their bounding boxes.



- `DOCUMENT_TEXT_DETECTION` also extracts text from an image, but the response is **optimized for dense text and documents**. The JSON includes page, block, paragraph, word, and break information.



Essential Tools for Cloud Platform

Google Cloud SDK is a set of libraries and tools that you can use to manage computing resources and applications hosted on Google Cloud Platform. With the Cloud SDK, you have [illegible] your deployments and more! The Cloud SDK also includes libraries for the Google Cloud APIs available in your language's package manager, like Maven, npm and NuGet.

VIEW CLOUD SDK

Simplify Your Cloud Management

Cloud Deployment Manager allows you to specify all the resources needed for your application in a declarative format using yaml. You can also use python or Jinja2 templates to parameterize the configuration and allow reuse of common deployment paradigms such as a load balanced, auto-scaled instance group. Treat your configuration as code and perform repeatable deployments.

VIEW DEPLOYMENT MANAGER

Collaborative Development on Git

Google Cloud Source Repositories are private Git repositories hosted on the Cloud Platform. You can use Cloud Source Repositories for collaborative development between members of your team, or use it with Stackdriver Debugger for realtime debugging of cloud apps. You can also connect your Cloud Source Repositories to repositories hosted on GitHub or Bitbucket. Connected repositories are synced automatically in both directions.

VIEW SOURCE REPOSITORIES

Learn more about `DOCUMENT_TEXT_DETECTION` for handwriting extraction (https://cloud.google.com/vision/docs/handwriting) and text extraction from files (PDF/TIFF) (https://cloud.google.com/vision/docs/pdf).

Text detection requests

Set up your GCP project and authentication

If you have not created a Google Cloud Platform (GCP) project (https://cloud.google.com/docs/overview/#projects) and service account credentials, do so now. Expand this section for instructions.

1. Sign in (https://accounts.google.com/Login) to your Google Account.

   If you don't already have one, sign up for a new account (https://accounts.google.com/SignUp).

2. Set up a Cloud Console project.

   SET UP A PROJECT

   Click to:

   - Create or select a project.
   - Enable the Cloud Vision API for that project.
   - Create a service account.
   - Download a private key as JSON.

   You can view and manage these resources at any time in the Cloud Console (https://console.cloud.google.com/).

3. Set the environment variable ***GOOGLE_APPLICATION_CREDENTIALS*** to the file path of the JSON file that contains your service account key. This variable only applies to your current shell session, so if you open a new session, set the variable again.

   Example: Linux or macOS

   Replace ***[PATH]*** with the file path of the JSON file that contains your service account key.

   ```
   export GOOGLE_APPLICATION_CREDENTIALS="[PATH]"
   ```

   For example:

   ```
   export GOOGLE_APPLICATION_CREDENTIALS="/home/user/Downloads/service-account-fil
   ```

   Example: Windows

   Replace ***[PATH]*** with the file path of the JSON file that contains your service account key, and ***[FILE_NAME]*** with the filename.

   With PowerShell:

   ```
   $env:GOOGLE_APPLICATION_CREDENTIALS="[PATH]"
   ```

For example:

```
$env:GOOGLE_APPLICATION_CREDENTIALS="C:\Users\username\Downloads\[FILE_NAME].js
```

With command prompt:

```
set GOOGLE_APPLICATION_CREDENTIALS=[PATH]
```

4. Install and initialize the Cloud SDK (https://cloud.google.com/sdk/docs/).

Detect text in a local image

The Vision API can perform feature detection on a local image file by sending the contents of the image file as a base64 encoded (https://cloud.google.com/vision/docs/base64) string in the body of your request.

REST & CMD LINE C# GO MORE

Before using any of the request data below, make the following replacements:

- ***base64-encoded-image***: The base64 representation (ASCII string) of your binary image data. This string should look similar to the following string:
  - `/9j/4QAYRXhpZgAA...9tAVx/zDQDlGxn//2Q==`

  Visit the base64 encode (https://cloud.google.com/vision/docs/base64) topic for more information.

HTTP method and URL:

```
POST https://vision.googleapis.com/v1/images:annotate
```

Request JSON body:

```
{
  "requests": [
    {
      "image": {
        "content": "base64-encoded-image"
      },
      "features": [
        {
          "type": "TEXT_DETECTION"
        }
      ]
    }
  ]
}
```

To send your request, choose one of these options:

CURL POWERSHELL

Note: If you are not executing the command below from Cloud Shell (https://cloud.google.com/shell/docs) or Compute Engine (https://cloud.google.com/compute/docs), ensure you have set the **`GOOGLE_APPLICATION_CREDENTIALS`** (https://cloud.google.com/docs/authentication/production) environment variable to your service account private key file path.

Save the request body in a file called `request.json`, and execute the following command:

```
curl -X POST \
-H "Authorization: Bearer "$(gcloud auth application-default print-access-token)
-H "Content-Type: application/json; charset=utf-8" \
-d @request.json \
https://vision.googleapis.com/v1/images:annotate
```

If the request is successful, the server returns a `200 OK` HTTP status code and the response in JSON format.

A `TEXT_DETECTION` response includes the detected phrase, its bounding box, and individual words and their bounding boxes.

Note: Zero coordinate values omitted. When the API detects a coordinate ("x" or "y") value of 0, ***that coordinate is omitted in the JSON response***. Thus, a response with a bounding poly around the entire image would be **[{},{"x": 100},{"x": 100,"y": 100},{"y": 100}] for an image that is 100px by 100px**. For more information, see the API Reference documentation (https://cloud.google.com/vision/docs/reference/rest/v1/images/annotate#boundingpoly).

Response

```
{
  "responses": [
    {
      "textAnnotations": [
        {
          "locale": "en",
          "description": "WAITING?\nPLEASE\nTURN OFF\nYOUR\nENGINE\n",
          "boundingPoly": {
            "vertices": [
              {
                "x": 341,
                "y": 828
              },
              {
                "x": 2249,
                "y": 828
              },
              {
                "x": 2249,
                "y": 1993
              },
              {
                "x": 341,
                "y": 1993
              }
            ]
          }
        },
        {
          "description": "WAITING?",
          "boundingPoly": {
            "vertices": [
              {
                "x": 352,
                "y": 828
              },
              {
                "x": 2248,
                "y": 911
              },
              {
                "x": 2238,
                "y": 1148
              },
              {
                "x": 342,
                "y": 1065
              }
            ]
          }
        },
        {
          "description": "PLEASE",
          "boundingPoly": {
            "vertices": [
              {
                "x": 1210,
                "y": 1233
              },
              {
                "x": 1907,
                "y": 1263
              },
              {
                "x": 1902,
                "y": 1383
              },
              {
                "x": 1205,
                "y": 1353
              }
            ]
          }
        },
        {
          "description": "TURN",
          "boundingPoly": {
            "vertices": [
              {
                "x": 1210,
                "y": 1418
              },
              {
                "x": 1730,
                "y": 1441
              },
              {
                "x": 1724,
                "y": 1564
              },
            {
              "x": 1205,
              "y": 1541
            }
          ]
        }
      },
      {
        "description": "OFF",
        "boundingPoly": {
          "vertices": [
            {
              "x": 1792,
              "y": 1443
            },
            {
              "x": 2128,
              "y": 1458
            },
            {
              "x": 2122,
              "y": 1581
            },
            {
              "x": 1787,
              "y": 1566
            }
          ]
        }
      },
      {
        "description": "YOUR",
        "boundingPoly": {
          "vertices": [
            {
              "x": 1219,
              "y": 1603
            },
            {
              "x": 1746,
              "y": 1629
            },
            {
              "x": 1740,
              "y": 1759
            },
            {
              "x": 1213,
              "y": 1733
            }
          ]
        }
      },
      {
        "description": "ENGINE",
        "boundingPoly": {
          "vertices": [
            {
              "x": 1222,
              "y": 1771
            },
            {
              "x": 1944,
              "y": 1834
            },
            {
              "x": 1930,
              "y": 1992
            },
            {
              "x": 1208,
              "y": 1928
            }
          ]
        }
      }
    ],
    "fullTextAnnotation": {
      "pages": [
                ...
                ]
              },
              "paragraphs": [
                    ...
                    ]
                  },
                  "words": [
                      ...
                      },
                      "symbols": [
                        ...
                      }
                    ]
                  }
                ],
                "blockType": "TEXT"
              },
              ...
            ]
          }
        ],
        "text": "WAITING?\nPLEASE\nTURN OFF\nYOUR\nENGINE\n"
      }
    }
  ]
}
```

Detect text in a remote image

For your convenience, the Vision API can perform feature detection directly on an image file located in Google Cloud Storage or on the Web without the need to send the contents of the image file in the body of your request.

Caution: When fetching images from HTTP/HTTPS URLs, Google cannot guarantee that the request will be completed. Your request may fail if the specified host denies the request (for example, due to request throttling or DOS (https://en.wikipedia.org/wiki/Denial-of-service_attack) prevention), or if Google throttles requests to the site for abuse prevention. **You should not depend on externally-hosted images for production applications**.

REST & CMD LINE   C#   MORE ▾

Before using any of the request data below, make the following replacements:

- ***cloud-storage-image-uri***: the path to a valid image file in a Cloud Storage bucket. You must at least have read privileges to the file. Example:
  - `gs://cloud-samples-data/vision/ocr/sign.jpg`

HTTP method and URL:

```
POST https://vision.googleapis.com/v1/images:annotate
```

Request JSON body:

```
{
  "requests": [
    {
      "image": {
        "source": {
          "imageUri": "cloud-storage-image-uri"
        }
       },
       "features": [
         {
           "type": "TEXT_DETECTION"
         }
       ]
    }
  ]
}
```

To send your request, choose one of these options:

CURL POWERSHELL

Note: If you are not executing the command below from Cloud Shell (https://cloud.google.com/shell/docs) or Compute Engine (https://cloud.google.com/compute/docs), ensure you have set the `GOOGLE_APPLICATION_CREDENTIALS` (https://cloud.google.com/docs/authentication/production) environment variable to your service account private key file path.

Save the request body in a file called `request.json`, and execute the following command:

```
curl -X POST \
-H "Authorization: Bearer "$(gcloud auth application-default print-access-token)
-H "Content-Type: application/json; charset=utf-8" \
-d @request.json \
https://vision.googleapis.com/v1/images:annotate
```

If the request is successful, the server returns a `200 OK` HTTP status code and the response in JSON format.

A `TEXT_DETECTION` response includes the detected phrase, its bounding box, and individual words and their bounding boxes.

> **Note: Zero coordinate values omitted.** When the API detects a coordinate ("x" or "y") value of 0, ***that coordinate is omitted in the JSON response***. Thus, a response with a bounding poly around the entire image would be
> **[{},{"x": 100},{"x": 100,"y": 100},{"y": 100}] for an image that is 100px by 100px**. For more information, see the API Reference documentation (https://cloud.google.com/vision/docs/reference/rest/v1/images/annotate#boundingpoly).

Response

```
{
  "responses": [
    {
      "textAnnotations": [
        {
          "locale": "en",
          "description": "WAITING?\nPLEASE\nTURN OFF\nYOUR\nENGINE\n",
          "boundingPoly": {
            "vertices": [
              {
                "x": 341,
                "y": 828
              },
              {
                "x": 2249,
                "y": 828
              },
              {
                "x": 2249,
                "y": 1993
              },
              {
                "x": 341,
                "y": 1993
              }
            ]
          }
        },
        {
          "description": "WAITING?",
          "boundingPoly": {
            "vertices": [
              {
                "x": 352,
                "y": 828
              },
              {
                "x": 2248,
                "y": 911
              },
              {
                "x": 2238,
                "y": 1148
              },
              {
                "x": 342,
                "y": 1065
              }
            ]
          }
        },
        {
          "description": "PLEASE",
          "boundingPoly": {
            "vertices": [
              {
                "x": 1210,
                "y": 1233
              },
              {
                "x": 1907,
                "y": 1263
              },
              {
                "x": 1902,
                "y": 1383
              },
              {
                "x": 1205,
                "y": 1353
              }
            ]
          }
        },
        {
          "description": "TURN",
          "boundingPoly": {
            "vertices": [
              {
                "x": 1210,
                "y": 1418
              },
              {
                "x": 1730,
                "y": 1441
              },
              {
                "x": 1724,
                "y": 1564
              },
              {
                "x": 1205,
                "y": 1541
              }
            ]
          }
        },
        {
          "description": "OFF",
          "boundingPoly": {
            "vertices": [
              {
                "x": 1792,
                "y": 1443
              },
              {
                "x": 2128,
                "y": 1458
              },
              {
                "x": 2122,
                "y": 1581
              },
              {
                "x": 1787,
                "y": 1566
              }
            ]
          }
        },
        {
          "description": "YOUR",
          "boundingPoly": {
            "vertices": [
              {
                "x": 1219,
                "y": 1603
              },
              {
                "x": 1746,
                "y": 1629
              },
              {
                "x": 1740,
                "y": 1759
              },
              {
                "x": 1213,
                "y": 1733
              }
            ]
          }
        },
        {
          "description": "ENGINE",
          "boundingPoly": {
            "vertices": [
              {
                "x": 1222,
                "y": 1771
              },
              {
                "x": 1944,
                "y": 1834
              },
              {
                "x": 1930,
                "y": 1992
              },
              {
                "x": 1208,
                "y": 1928
              }
            ]
          }
        }
```

```
      ],
      "fullTextAnnotation": {
        "pages": [
                  ...
                  ]
                },
                "paragraphs": [
                      ...
                      ]
                    },
                    "words": [
                        ...
                        },
                        "symbols": [
                        ...
                      }
                    ]
                  }
                ],
                "blockType": "TEXT"
              },
              ...
            ]
          }
        ],
        "text": "WAITING?\nPLEASE\nTURN OFF\nYOUR\nENGINE\n"
      }
    }
  ]
}
```

Specify the language (optional)

Both types of OCR requests support one or more `languageHints` that specify the language of any text in the image. However, in most cases, an empty value yields the best results since it enables automatic language detection. For languages based on the Latin alphabet, setting `languageHints` is not needed. In rare cases, when the language of the text in the image is known, setting a hint helps get better results (although it can be a significant hindrance if the hint is wrong). Text detection returns an error if one or more of the specified languages is not one of the supported languages (https://cloud.google.com/vision/docs/languages).

If you choose to provide a language hint, modify the body of your request (`request.json` file) to provide the string of one of the supported languages in the `imageContext.languageHints` field as shown below:

```
{
  "requests": [
    {
      "image": {
        "source": {
          "imageUri": "image-url"
        }
      },
      "features": [
        {
          "type": "DOCUMENT_TEXT_DETECTION"
        }
      ],
      "imageContext": {
        "languageHints": ["en-t-i0-handwrit"]
      }
    }
  ]
}
```

How do language hints work?

The `languageHint` format follows the BCP47 (https://tools.ietf.org/html/bcp47) language code formatting guidelines. The BCP47 specified format is as follows:

`language` ["-" `script`] ["-" `region`] *("-" `variant`) *("-" `extension`) ["-" `privateuse`].

For example, the language hint "`en`-`t`-`i0`-`handwrit`" specifies English language (`en`), transform (https://tools.ietf.org/html/rfc6497) extension singleton (`t`), input method engine transform (http://www.unicode.org/reports/tr35/) extension code (`i0`), and handwriting transform code (http://unicode.org/repos/cldr/tags/latest/common/bcp47/transform_ime.xml) (`handwrit`). This roughly says the language is "English transformed from handwriting." You do not need to specify a script code because `Latn` is implied by the "`en`" language.

Multi-regional support

This functionality currently only applies to the OCR feature (types `TEXT_DETECTION` or `DOCUMENT_TEXT_DETECTION`).

You can now specify continent-level data storage and OCR processing. The following regions are currently supported:

- `us`: USA country only
- `eu`: The European Union

Locations

Cloud Vision offers you some control over where the resources for your project are stored and processed. In particular, you can configure Cloud Vision to store and process your data only in the European Union.

By default Cloud Vision stores and processes resources in a **Global** location, which means that Cloud Vision doesn't guarantee that your resources will remain within a particular location or region. If you choose the **European Union** location, Google will store your data and process it only in the European Union. You and your users can access the data from any location.

Setting the location using the API

Cloud Vision supports both a global API endpoint (`vision.googleapis.com`) and a European Union endpoint (`eu-vision.googleapis.com`). To store and process your data in the European Union only, use the URI `eu-vision.googleapis.com` in place of `vision.googleapis.com` for your REST API calls:

- `https://eu-vision.googleapis.com/v1/images:annotate`
- `https://eu-vision.googleapis.com/v1/images:asyncBatchAnnotate`
- `https://eu-vision.googleapis.com/v1/files:annotate`
- `https://eu-vision.googleapis.com/v1/files:asyncBatchAnnotate`

Setting the location using the client libraries

The Vision API client libraries accesses the global API endpoint (`vision.googleapis.com`) by default. To store and process your data in the European Union only, you need to explicitly set the endpoint (`eu-vision.googleapis.com`). The code samples below show how to configure this setting.

REST & CMD LINE C# GO MORE ▾

Before using any of the request data below, make the following replacements:

- ***cloud-storage-image-uri***: the path to a valid image file in a Cloud Storage bucket. You must at least have read privileges to the file. Example:
    - `gs://cloud-samples-data/vision/ocr/sign.jpg`

HTTP method and URL:

```
POST https://eu-vision.googleapis.com/v1/images:annotate
```

Request JSON body:

```
{
  "requests": [
    {
      "image": {
        "source": {
          "imageUri": "cloud-storage-image-uri"
        }
      },
      "features": [
        {
          "type": "TEXT_DETECTION"
        }
      ]
    }
  ]
}
```

To send your request, choose one of these options:

CURL POWERSHELL

Note: If you are not executing the command below from Cloud Shell (https://cloud.google.com/shell/docs) or Compute Engine (https://cloud.google.com/compute/docs), ensure you have set the `GOOGLE_APPLICATION_CREDENTIALS` (https://cloud.google.com/docs/authentication/production) environment variable to your service account private key file path.

Save the request body in a file called `request.json`, and execute the following command:

```
curl -X POST \
-H "Authorization: Bearer "$(gcloud auth application-default print-access-token)
-H "Content-Type: application/json; charset=utf-8" \
-d @request.json \
https://eu-vision.googleapis.com/v1/images:annotate
```

If the request is successful, the server returns a `200 OK` HTTP status code and the response in JSON format.

A `TEXT_DETECTION` response includes the detected phrase, its bounding box, and individual words and their bounding boxes.

Note: Zero coordinate values omitted. When the API detects a coordinate ("x" or "y") value of 0, ***that coordinate is omitted in the JSON response***. Thus, a response with a bounding poly around the entire image would be **[{},{"x": 100},{"x": 100,"y": 100},{"y": 100}] for an image that is 100px by 100px**. For more information, see the API Reference documentation (https://cloud.google.com/vision/docs/reference/rest/v1/images/annotate#boundingpoly).

Response

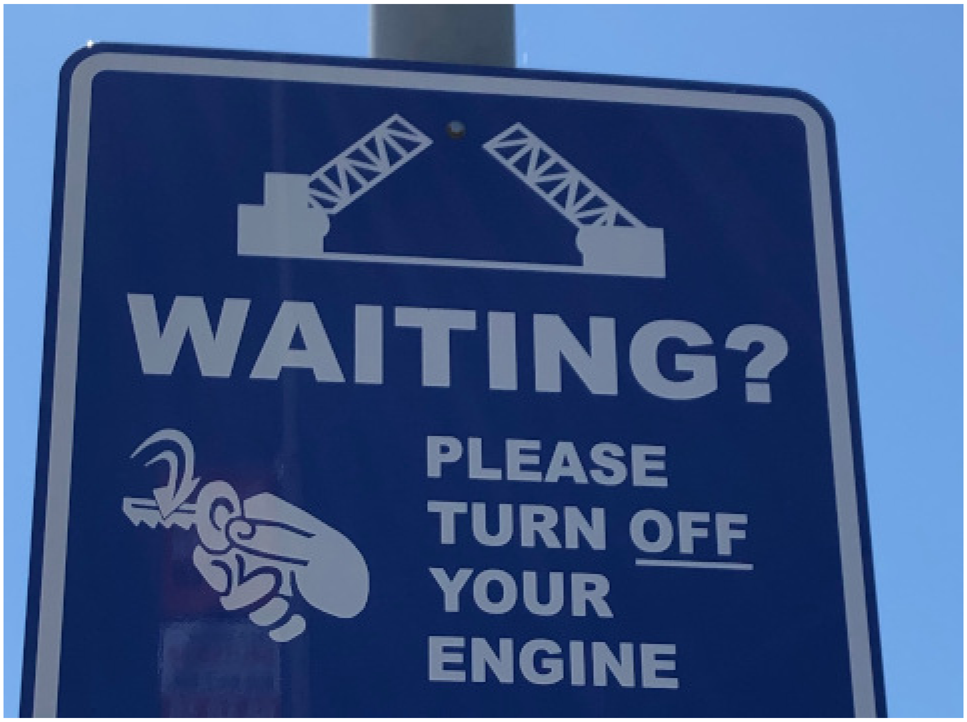
```
{
  "responses": [
    {
      "textAnnotations": [
        {
          "locale": "en",
          "description": "WAITING?\nPLEASE\nTURN OFF\nYOUR\nENGINE\n",
          "boundingPoly": {
            "vertices": [
              {
                "x": 341,
                "y": 828
              },
              {
                "x": 2249,
                "y": 828
              },
              {
                "x": 2249,
                "y": 1993
              },
              {
                "x": 341,
                "y": 1993
              }
            ]
          }
        },
        {
          "description": "WAITING?",
          "boundingPoly": {
            "vertices": [
              {
                "x": 352,
                "y": 828
              },
              {
                "x": 2248,
                "y": 911
              },
              {
                "x": 2238,
                "y": 1148
              },
              {
                "x": 342,
                "y": 1065
              }
            ]
          }
        },
        {
          "description": "PLEASE",
          "boundingPoly": {
            "vertices": [
              {
                "x": 1210,
                "y": 1233
              },
              {
                "x": 1907,
                "y": 1263
              },
              {
                "x": 1902,
                "y": 1383
              },
              {
                "x": 1205,
                "y": 1353
              }
            ]
          }
        },
        {
          "description": "TURN",
          "boundingPoly": {
            "vertices": [
              {
                "x": 1210,
                "y": 1418
              },
              {
                "x": 1730,
                "y": 1441
              },
              {
                "x": 1724,
                "y": 1564
              },
              {
                "x": 1205,
                "y": 1541
              }
            ]
          }
        },
        {
          "description": "OFF",
          "boundingPoly": {
            "vertices": [
              {
                "x": 1792,
                "y": 1443
              },
              {
                "x": 2128,
                "y": 1458
              },
              {
                "x": 2122,
                "y": 1581
              },
              {
                "x": 1787,
                "y": 1566
              }
            ]
          }
        },
        {
          "description": "YOUR",
          "boundingPoly": {
            "vertices": [
              {
                "x": 1219,
                "y": 1603
              },
              {
                "x": 1746,
                "y": 1629
              },
              {
                "x": 1740,
                "y": 1759
              },
              {
                "x": 1213,
                "y": 1733
              }
            ]
          }
        },
        {
          "description": "ENGINE",
          "boundingPoly": {
            "vertices": [
              {
                "x": 1222,
                "y": 1771
          },
          {
            "x": 1944,
            "y": 1834
          },
          {
            "x": 1930,
            "y": 1992
          },
          {
            "x": 1208,
            "y": 1928
          }
        ]
      }
    }
  ],
  "fullTextAnnotation": {
    "pages": [
              ...
              ]
            },
            "paragraphs": [
                  ...
                  ]
                },
                "words": [
                    ...
                    },
                    "symbols": [
                    ...
                  }
                ]
              }
            ],
            "blockType": "TEXT"
          },
          ...
        ]
      }
    ],
    "text": "WAITING?\nPLEASE\nTURN OFF\nYOUR\nENGINE\n"
  }
}
```

```
  ]
}
```

Try it

Try text detection and document text detection below. You can use the image specified already (`gs://cloud-samples-data/vision/ocr/sign.jpg`) by clicking **Execute**, or you can specify your own image in its place.

To try document text detection, update the value of `type` to `DOCUMENT_TEXT_DETECTION`.



Last updated January 10, 2020.