Shot Change Detection Tutorial

Audience

This tutorial is designed to let you quickly start exploring and developing applications with the Video Intelligence API. It is designed for people familiar with basic programming, though even without much programming knowledge, you should be able to follow along. Having walked through this tutorial, you should be able to use the Reference documentation (https://cloud.google.com/video-intelligence/docs/reference/rest/) to create your own basic applications.

This tutorial steps through a Video Intelligence API application using Python code. The purpose here is not to explain the Python client libraries, but to explain how to make calls to the Video Intelligence API. Applications in Java and Node.js are essentially similar.

If you're looking for a code-only example or an example in another language, check out the companion how-to guide (https://cloud.google.com/video-intelligence/docs/analyze-shots).

Prerequisites

This tutorial has several prerequisites:

- You've set up a Cloud Video Intelligence API project (https://cloud.google.com/video-intelligence/docs/before-you-begin) in the Google Cloud Console.
- You've set up your environment using a service account and Application Default Credentials (https://cloud.google.com/video-intelligence/docs/common/auth#set_up_a_service_account).
- You have basic familiarity with Python (https://www.python.org/) programming.
- Set up your Python development environment. It is recommended that you have the latest version of Python, `pip`, and `virtualenv` installed on your system. For instructions, refer to

the Python Development Environment Setup Guide (https://cloud.google.com/python/setup) for Google Cloud Platform.

- You've installed the Google Cloud client library (https://github.com/GoogleCloudPlatform/google-cloud-python/tree/master/videointelligence)

Annotating a video using Shot Change detection

This tutorial walks you through a basic Video API application, using a `SHOT_CHANGE_DETECTION` request. A `SHOT_CHANGE_DETECTION` request provides the annotation results:

- List of all shots that occur within the video
- For each shot, provide the start and end time of the shot

We'll show the entire code first. (Note that we have removed most comments from this code in order to show you how brief it is. We'll provide more comments as we walk through the code.)

video/cloud-client/shotchange/shotchange.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/shotchange/shotchange.py)

)RM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/VIDEO/CLOUD-CLIENT/SHOTCHANGE/SHOTCHANGE.PY)

```
import argparse

from google.cloud import videointelligence


def analyze_shots(path):
    """ Detects camera shot changes. """
    video_client = videointelligence.VideoIntelligenceServiceClient()
    features = [videointelligence.enums.Feature.SHOT_CHANGE_DETECTION]
    operation = video_client.annotate_video(path, features=features)
    print('\nProcessing video for shot change annotations:')

    result = operation.result(timeout=120)
    print('\nFinished processing.')

    for i, shot in enumerate(result.annotation_results[0].shot_annotations):
        start_time = (shot.start_time_offset.seconds +
                      shot.start_time_offset.nanos / 1e9)
        end_time = (shot.end_time_offset.seconds +
```

```
                    shot.end_time_offset.nanos / 1e9)
        print('\tShot {}: {} to {}'.format(i, start_time, end_time))


if __name__ == '__main__':
    parser = argparse.ArgumentParser(
        description=__doc__,
        formatter_class=argparse.RawDescriptionHelpFormatter)
    parser.add_argument('path', help='GCS path for shot change detection.')
    args = parser.parse_args()

    analyze_shots(args.path)
```

This simple application performs the following tasks:

- Imports the libraries necessary to run the application
- Takes a video file stored in Google Cloud Storage URI as an argument and passes it to the `main()` function
- Gets credentials to run the Video Intelligence API service
- Creates a video annotation request to send to the video service
- Sends the request and returns a long running operation
- Loops over the long running operation until the video is processed and return values are available
- Parses the response for the service and displays it to the user

We'll go over these steps in more detail below.

Importing libraries

video/cloud-client/shotchange/shotchange.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/shotchange/shotchange.py)

ORM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/VIDEO/CLOUD-CLIENT/SHOTCHANGE/SHOTCHANGE.PY)

```
import argparse

from google.cloud import videointelligence
```

We import `argparse` to allow the application to accept input filenames as arguments.

For using the Cloud Video Intelligence API, we also import the `google.cloud.videointelligence` library, which holds the directory of our API calls and enumeration constants.

Running Your application

video/cloud-client/shotchange/shotchange.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/shotchange/shotchange.py)

ORM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/VIDEO/CLOUD-CLIENT/SHOTCHANGE/SHOTCHANGE.PY)

```
parser = argparse.ArgumentParser(
    description=__doc__,
    formatter_class=argparse.RawDescriptionHelpFormatter)
parser.add_argument('path', help='GCS path for shot change detection.')
args = parser.parse_args()

analyze_shots(args.path)
```

Here, we parse the passed argument for the Google Cloud Storage URI of the video filename and pass it to the `main()` function.

Authenticating to the API

Before communicating with the Video Intelligence API service, you need to authenticate your service using previously acquired credentials. Within an application, the simplest way to obtain credentials is to use Application Default Credentials (https://cloud.google.com/video-intelligence/docs/common/auth#adc) (ADC). By default, ADC will attempt to obtain credentials from the `GOOGLE_APPLICATION_CREDENTIALS` environment file, which should be set to point to your service account's JSON key file. (You should have set up your service account and environment to use ADC in the Quickstart (https://cloud.google.com/video-intelligence/docs/getting-started). See Setting Up a Service Account (https://cloud.google.com/video-intelligence/docs/common/auth#set_up_a_service_account) for more information.)

Constructing the request

video/cloud-client/shotchange/shotchange.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/shotchange/shotchange.py)

ORM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/VIDEO/CLOUD-CLIENT/SHOTCHANGE/SHOTCHANGE.PY)

```
video_client = videointelligence.VideoIntelligenceServiceClient()
features = [videointelligence.enums.Feature.SHOT_CHANGE_DETECTION]
operation = video_client.annotate_video(path, features=features)
```

Now that our Video Intelligence API service is ready, we can construct a request to that service. Requests to the Video Intelligence API are provided as JSON objects. See the Video Intelligence API Reference (https://cloud.google.com/video-intelligence/docs/reference/rest) for complete information on the specific structure of such a request.

This code snippet performs the following tasks:

1. Constructs the JSON for a POST request to the `annotate_video()` method.
2. Injects the Google Cloud Storage location of our passed video filename into the request.
3. Indicates that the `annotate` method should perform a `SHOT_CHANGE_DETECTION`.

Constructing the Long Running operation

When we first execute a request against the Video Intelligence API, we do not get an immediate result; instead we get an *operation name*, stored within the response's `name` field, which we can then use to check for results at a later time.

Passing that operation's name (which is a numerical string) to the Video Intelligence API's *Operations Service* `operations` method returns the current state of the operation. A sample response is shown below:

```
{
   "response":{
      "@type":"type.googleapis.com/google.cloud.videointelligence.v1.AnnotateVideoRe
   },
   "name":"us-west1.17159971042783089144",
   "metadata":{
```

```
      "annotationProgress":[
         {
            "inputUri":"/demomaker/gbikes_dinosaur.mp4",
            "updateTime":"2017-01-27T19:45:54.297807Z",
            "startTime":"2017-01-27T19:45:54.275023Z"
         }
      ],
      "@type":"type.googleapis.com/google.cloud.videointelligence.v1.AnnotateVideoPr
   }
}
```

Note that the `response` field at this time only contains an `@type` field, denoting the type of that response. Once results are actually available, the response field will contain results of that type.

Checking the operation

video/cloud-client/shotchange/shotchange.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/shotchange/shotchange.py)

ORM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/VIDEO/CLOUD-CLIENT/SHOTCHANGE/SHOTCHANGE.PY)

```
result = operation.result(timeout=120)
print('\nFinished processing.')
```

Using the existing operation request for our existing operation, we construct a `while` loop to periodically check the state of that operation. Once our operation has indicated that the operation is `done`, we break out of the loop and can parse the response.

Parsing the response

video/cloud-client/shotchange/shotchange.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/shotchange/shotchange.py)

ORM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/VIDEO/CLOUD-CLIENT/SHOTCHANGE/SHOTCHANGE.PY)

```
for i, shot in enumerate(result.annotation_results[0].shot_annotations):
    start_time = (shot.start_time_offset.seconds +
```

```
                shot.start_time_offset.nanos / 1e9)
    end_time = (shot.end_time_offset.seconds +
                shot.end_time_offset.nanos / 1e9)
    print('\tShot {}: {} to {}'.format(i, start_time, end_time))
```

Once the operation has been completed, the response will contain an AnnotateVideoResponse (https://cloud.google.com/video-intelligence/docs/reference/rest/Shared.Types/AnnotateVideoResponse), which consists of a list of **`annotationResults`**, one for each video sent in the request. Because we sent only one video in the request, we take the first **`shotAnnotations`** of the result. We walk through all the 'segments' for the video.

Running our application

To run our application, simply pass it the Google Cloud Storage URI of a video:

```
$ python shotchange.py gs://demomaker/gbikes_dinosaur.mp4
operationId=us-west1.12468772851081463748
Operation us-west1.12468772851081463748 started: 2017-01-30T01:53:45.981043Z
Processing video for shot change annotations:
Finished processing.
  Shot 0: 0.0 to 5.166666
  Shot 1: 5.233333 to 10.066666
  Shot 2: 10.1 to 28.133333
  Shot 3: 28.166666 to 42.766666
```

Congratulations! You've performed an annotation task using the Video Intelligence API!
