clarifai Documentation

Release 2.3.0

Clarifai

Aug 21, 2018

First steps

CHAPTER 1

First steps

1.1 Installation Guide

Note: Generally, the python client works with Python 2.x and Python 3.x. But it is only tested against 2.7 and 3.5. Feel free to report BUGs if you encounter on other versions.

1.1.1 Install the package

You can install the latest stable clarifai using pip (which is the canonical way to install Python packages). To install using `pip` run:

```
pip install clarifai --upgrade
```

You can also install from git using latest source:

```
pip install git+git://github.com/Clarifai/clarifai-python.git
```

1.1.2 Configuration

The client uses CLARIFAI_API_KEY for authentication. Each application you create uses its own unique ID and secret to authenticate requests. The client will use the authentication information passed to it by one of three methods with the following precedence order:

1. Passed in to the constructor through the `api_key` parameter.
2. Set as the CLARIFAI_API_KEY environment variable.
3. Placed in the .clarifai/config file using the command below.

You can get these values from https://developer.clarifai.com/account/applications and then run:

```
$ clarifai config
CLARIFAI_API_KEY: []: ************************************YQEd
```

If you do not see any error message after this, you are all set and can proceed with using the client.

1.1.3 Windows Users

For Windows users, you may fail running the `clarifai config` when you try to configure the runtime environment. This is because Windows uses the file extension to determine executables and by default file `clarifai` without file extension is nonexecutable. In order to run the command, you may want to launch it with the python interpreter.

```
C:\Python27>python.exe Scripts\clarifai config
CLARIFAI_API_KEY: []: ************************************YQEd
```

1.1.4 AWS Lambda Users

For AWS Lambda users, in order to use the library correctly, you are recommended to set two environmental variables *CLARIFAI_API_KEY* in the lambda function configuration, or hardcode the APP_ID and APP_SECRET in the API instantiation.

1.2 Tutorial

Each of the examples below is a small independent code snippet within 10 lines that could work by copy and paste to a python source code file. By playing with them, you should be getting started with Clarifai API. For more information about the API, check the API Reference.

1.2.1 Predict Tutorial

Predict with Public Models

For more information on any of the public models, visit https://developer.clarifai.com/models

```
from clarifai.rest import ClarifaiApp

app = ClarifaiApp()

#General model
model = app.models.get('general-v1.3')

response = model.predict_by_url(url='https://samples.clarifai.com/metro-north.jpg')


#Travel model
model = app.models.get('travel-v1.0')

response = model.predict_by_url(url='https://samples.clarifai.com/travel.jpg')


#Food model
model = app.models.get('food-items-v1.0')
```

(continues on next page)

(continued from previous page)

```
response = model.predict_by_url(url='https://samples.clarifai.com/food.jpg')


#NSFW model
model = app.models.get('nsfw-v1.0')

response = model.predict_by_url(url='https://samples.clarifai.com/nsfw.jpg')


#Apparel model
model = app.models.get('apparel')

response = model.predict_by_url(url='https://samples.clarifai.com/apparel.jpg')


#Celebrity model
model = app.models.get('celeb-v1.3')

response = model.predict_by_url(url='https://samples.clarifai.com/celebrity.jpg')


#Demographics model
model = app.models.get('demographics')

response = model.predict_by_url(url='https://samples.clarifai.com/demographics.jpg')


#Face Detection model
model = app.models.get('face-v1.3')

response = model.predict_by_url(url='https://developer.clarifai.com/static/images/
↪model-samples/face-001.jpg')


#Focus Detection model
model = app.models.get('focus')

response = model.predict_by_url(url='https://samples.clarifai.com/focus.jpg')


#General Embedding model
model = app.models.get('general-v1.3', model_type='embed')

response = model.predict_by_url(url='https://samples.clarifai.com/metro-north.jpg')


#Logo model
model = app.models.get('logo')

response = model.predict_by_url(url='https://samples.clarifai.com/logo.jpg')


#Color model
model = app.models.get('color', model_type='color')

response = model.predict_by_url(url='https://samples.clarifai.com/wedding.jpg')
```

1.2.2 Feedback Tutorial

Concept model prediction

```
from clarifai.rest import ClarifaiApp
from clarifai.rest import FeedbackInfo

app = ClarifaiApp()

# positive feedback: this is a dog
m = app.models.get('general-v1.3')

m.send_concept_feedback(input_id='id1', url='https://samples.clarifai.com/dog2.jpeg',
                        concepts=['dog', 'animal'],
                        feedback_info=FeedbackInfo(output_id='OID',
                                                   session_id='SID',
                                                   end_user_id='UID',
                                                   event_type='annotation'))

# negative feedback: this is not a cat
m = app.models.get('general-v1.3')

m.send_concept_feedback(input_id='id1', url='https://samples.clarifai.com/dog2.jpeg',
                        not_concepts=['cat', 'kitty'],
                        feedback_info=FeedbackInfo(output_id='OID',
                                                   session_id='SID',
                                                   end_user_id='UID',
                                                   event_type='annotation'))

# all together: this is a dog but not a cat
m = app.models.get('general-v1.3')

m.send_concept_feedback(input_id='id1', url='https://samples.clarifai.com/dog2.jpeg',
                        concepts=['dog'], not_concepts=['cat', 'kitty'],
                        feedback_info=FeedbackInfo(output_id='OID',
                                                   session_id='SID',
                                                   end_user_id='UID',
                                                   event_type='annotation'))
```

Detection model prediction

```
from clarifai.rest import ClarifaiApp
from clarifai.rest import FeedbackInfo
from clarifai.rest import Region, RegionInfo, BoundingBox, Concept

app = ClarifaiApp()

m.send_region_feedback(input_id='id2', url='https://developer.clarifai.com/static/
↪images/model-samples/celeb-001.jpg',
                       regions=[Region(region_info=RegionInfo(bbox=BoundingBox(top_
↪row=0.1,
                                                                               left_
↪col=0.2,
                                                                               bottom_
↪row=0.5,
```

(continues on next page)

(continued from previous page)

```
                                                                              right_
↪col=0.5)),
                                       concepts=[Concept(concept_id='people',␣
↪value=True),
                                                 Concept(concept_id='portrait',␣
↪value=True)])],
                       feedback_info=FeedbackInfo(output_id='OID',
                                                  session_id='SID',
                                                  end_user_id='UID',
                                                  event_type='annotation'))
```

Face detection model prediction

send feedback for celebrity model

```
from clarifai.rest import ClarifaiApp
from clarifai.rest import FeedbackInfo
from clarifai.rest import Region, RegionInfo, BoundingBox, Concept
from clarifai.rest import Face, FaceIdentity
from clarifai.rest import FaceAgeAppearance, FaceGenderAppearance,␣
↪FaceMulticulturalAppearance

app = ClarifaiApp()

#
# send feedback for celebrity model
#
m.send_region_feedback(input_id='id2', url='https://developer.clarifai.com/static/
↪images/model-samples/celeb-001.jpg',
                       regions=[Region(region_info=RegionInfo(bbox=BoundingBox(top_
↪row=0.1,
                                                                              left_
↪col=0.2,
                                                                              bottom_
↪row=0.5,
                                                                              right_
↪col=0.5)),
                                                           ␣
↪face=Face(identity=FaceIdentity([Concept(concept_id='celeb1', value=True)]))
                                       )
                                ],
                       feedback_info=FeedbackInfo(output_id='OID',
                                                  session_id='SID',
                                                  end_user_id='UID',
                                                  event_type='annotation'))

#
# send feedback for age, gender, multicultural appearance
#

m.send_region_feedback(input_id='id2', url='https://developer.clarifai.com/static/
↪images/model-samples/celeb-001.jpg',
                       regions=[Region(region_info=RegionInfo(bbox=BoundingBox(top_
↪row=0.1,
                                                                              left_
↪col=0.2,
```

(continues on next page)

(continued from previous page)

```
                                                                                    bottom_
↪row=0.5,
                                                                                    right_
↪col=0.5)),
                                       face=Face(age_
↪appearance=FaceAgeAppearance([Concept(concept_id='20', value=True),
                                                                                    ␣
↪Concept(concept_id='30', value=False)
                                                                                     ]),
                                                 gender_
↪appearance=FaceGenderAppearance([Concept(concept_id='male', value=True)]),
                                                 multicultural_
↪appearance=FaceMulticulturalAppearance([Concept(concept_id='asian', value=True)])
                                                )
                                       )
                                ],
                       feedback_info=FeedbackInfo(output_id='OID',
                                                  session_id='SID',
                                                  end_user_id='UID',
                                                  event_type='annotation'))
```

1.2.3 Upload Images

```python
from clarifai.rest import ClarifaiApp

app = ClarifaiApp()

app.inputs.create_image_from_url(url='https://samples.clarifai.com/puppy.jpeg',␣
↪concepts=['my puppy'])
app.inputs.create_image_from_url(url='https://samples.clarifai.com/wedding.jpg', not_
↪concepts=['my puppy'])
```

1.2.4 Create a Model

Note: This assumes you follow through the tutorial and finished the "Upload Images" Otherwise you may not be able to create the model.

```python
model = app.models.create(model_id="puppy", concepts=["my puppy"])
```

1.2.5 Train the Model

Note: This assumes you follow through the tutorial and finished the "Upload Images" and "Create a Model" to create a model. Otherwise you may not be able to train the model.

```python
model.train()
```

1.2.6 Predict with Model

Note: This assumes you follow through the tutorial and finished the "Upload Images", "Create a Model", and "Train the Model". Otherwise you may not be able to make predictions with the model.

```python
from clarifai.rest import ClarifaiApp

app = ClarifaiApp()

model = app.models.get('puppy')
model.predict_by_url('https://samples.clarifai.com/metro-north.jpg')
```

1.2.7 Instantiate an Image

```python
from clarifai.rest import Image as ClImage

# make an image with an url
img = ClImage(url='https://samples.clarifai.com/dog1.jpeg')

# make an image with a filename
img = ClImage(filename='/tmp/user/dog.jpg')

# allow duplicate url
img = ClImage(url='https://samples.clarifai.com/dog1.jpeg', allow_dup_url=True)

# make an image with concepts
img = ClImage(url='https://samples.clarifai.com/dog1.jpeg', \
              concepts=['cat', 'animal'])

# make an image with metadata
img = ClImage(url='https://samples.clarifai.com/dog1.jpeg', \
              concepts=['cat', 'animal'], \
              metadata={'id':123,
                        'city':'New York'
                       })
```

1.2.8 Bulk Import Images

If you have a large amount of images, you may not want to upload them one by one by calling *app.inputs.create_image_from_url('https://samples.clarifai.com/dog1.jpeg')*

Instead you may want to use the bulk import API.

Note: The max number images per batch is 128. If you have more than 128 images to upload, you may want to chunk them into 128 or less, and bulk import them batch by batch.

In order to use this, you have to instantiate Image() objects from various sources.

```python
from clarifai.rest import ClarifaiApp
from clarifai.rest import Image as ClImage
```

(continues on next page)

(continued from previous page)

```python

# assume there are 100 urls in the list
images = []
for url in urls:
  img = ClImage(url=url)
  images.append(img)

app.inputs.bulk_create_images(images)
```

1.2.9 Search the Image

Note: This assumes you follow through the tutorial and finished the "Upload Images" Otherwise you may not be able to search

```python
from clarifai.rest import ClarifaiApp

app = ClarifaiApp()

app.inputs.search_by_annotated_concepts(concept='my puppy')

app.inputs.search_by_predicted_concepts(concept='dog')

app.inputs.search_by_image(url='https://samples.clarifai.com/dog1.jpeg')

app.inputs.search_by_metadata(metadata={'key':'value'})
```

1.3 Basic Concepts

This page lists a few basic notions used in the Clarifai API.

1.3.1 Image

Image is a straightforward notion. It represents a picture in a digital format.

In the Clarifai API, an image could be represented by a url, a local filename, raw bytes of the image, or bytes encoded in base64. To construct a new Image object, use the Image() constructor.

1.3.2 Video

In the Clarifai API, Video is considered a sequence of frames where each frame represents one second of the video. This means that after running prediction, the models will return results for every second of the video.

Video is used similarly to Image. Video can be represented by a url, a local filename, raw bytes of the video, or bytes encoded in base64. To construct a new Video object, use the Video() constructor.

1.3.3 Input

Input is a more general notion; it can be either Image or Video. Input is used for uploading, prediction, search, etc.

Each Input has a unique ID and can be associated with Concepts.

1.3.4 Concept

Concept represents a word to associate with Inputs. Concept can be a concrete notion like “dog” or “cat”, or a abstract notion like “love”. All models of type *concept* return a set of Concepts in the prediction for the given Input. Each Concept has a unique ID, and a name (in the data science nomenclature also sometimes referred to as a label, or a tag).

1.3.5 Model

Model is a machine learning algorithm that takes Input, such as Image or Video, runs a prediction, and outputs the results. There are several types of models: Concept Model, Face Detection Model, Color Model, etc. They all return different types of results. For example, Concept Model returns associated Concepts for each Input, Face Detection Model returns the locations of faces, Color Model returns dominant colors, etc.

There are public (pre-defined) models you can use, and custom models that you train yourself with your own Inputs.

1.3.6 Custom models

Custom models must be of type *concept*.

When creating a custom model, you provide it with your Inputs and associated Concepts. After being trained, the model can predict what Concepts are associated with never-before-seen Inputs, together with probabilities for confidence of the prediction.

Models can also be evaluated for measuring their prediction capability.

1.3.7 Image Crop

Image crop is defined as a crop box within an image. We use this in visual search so user does not have to crop an image before the search.

We use percentage coordinates instead of pixel coordinates to specify the crop box.

A four-element-tuple represents a crop box, in (top_y, left_x, bottom_y, right_x) order.

So a (0.3, 0.2, 0.6, 0.4) represents a box horizontally spanning from 20%-40% and vertically spanning 30%-60% of the image, measured from the top left corner.

1.3.8 Workflow

Workflow enables you to run prediction on Inputs using several Models in one request.

Installation Guide Install Clarifai python library

Tutorial Getting started with Clarifai API using python client

Basic Concepts Getting familiar with basic concepts used in Clarifai API

contributing Contributing with a pull request

1.4 API Reference

This is the API Reference documentation extracted from the source code.

1.4.1 Application

1.4.2 Concepts

1.4.3 Inputs

1.4.4 Models

1.4.5 Model

1.4.6 Search Syntax

1.4.7 Exceptions