eyewitness Documentation

Release 1.2.1

ching-hua, yang

Jan 29, 2020

Contents

1 Contents **3**

1.1 Quick Start . . . 3
1.2 ImageId . . . 7
1.3 Image Utils . . . 8
1.4 Object Detector . . . 10
1.5 Detection Utils . . . 10
1.6 Detection Result Filter . . . 11
1.7 Evaluation Utils . . . 12
1.8 A BboxMAPEvaluator Example . . . 12
1.9 Audience Id . . . 12
1.10 Feedback Msg Utils . . . 13
1.11 Handler Example . . . 14
1.12 ORM DB Models . . . 16
1.13 DataSet Utils . . . 18
1.14 MOT Module . . . 20

Python Module Index **23**

Index **25**

Eyewitness is light weight framework for object detection application

naive schema

CHAPTER 1

Contents

1.1 Quick Start

a naive fake example including:

- ImageProducer (generate image)
- ObjectDetector (generate detection result: detected objects on the image)
- DetectionResultHandler(write the detection result to db)

we will provide a fake image producer, object_detector in following code.

1.1.1 Pre-requirement

install eyewitness

```
pip install eyewitness
```

download the pikachu image as pikachu.png

```
wget -O pikachu.png https://upload.wikimedia.org/wikipedia/en/a/a6/Pok%C3%A9mon_
↪Pikachu_art.png
```

1.1.2 Implement a pikachu ImageProducer

keep yielding a pikachu image

```
import time
import arrow
from eyewitness.image_id import ImageId
from eyewitness.image_utils import Image
from eyewitness.config import IN_MEMORY
from eyewitness.image_utils import ImageHandler, ImageProducer

class InMemoryImageProducer(ImageProducer):
    def __init__(self, image_path, channel='pikachu', interval_s=3):
        self.pikachu =  ImageHandler.read_image_file(image_path)
        self.interval_s = interval_s
        self.channel = channel

    def produce_method(self):

        return IN_MEMORY

    def produce_image(self):
        while True:
            image_id = ImageId(channel=self.channel, timestamp=arrow.now().timestamp,␣
↪file_format='png')
            image_obj = Image(image_id, pil_image_obj=self.pikachu)
            yield image_obj
            time.sleep(self.interval_s)
```

1.1.3 Implement a fake ObjectDetector

always detect and draw bbox at same place

```
import os
from pathlib import Path

from eyewitness.object_detector import ObjectDetector
from eyewitness.detection_utils import DetectionResult
from eyewitness.config import (
   BBOX,
   BoundedBoxObject,
   DRAWN_IMAGE_PATH,
   IMAGE_ID,
   DETECTED_OBJECTS,
   DETECTION_METHOD
)

class FakePikachuDetector(ObjectDetector):
   def __init__(self, enable_draw_bbox=True):
      self.enable_draw_bbox = enable_draw_bbox

   def detect(self, image_obj):
      """
      fake detect method for FakeObjDetector

      Parameters
      ----------
      image_obj: eyewitness.image_utils.Image

      Returns
      -------
      DetectionResult

      """
      image_dict = {
            IMAGE_ID: image_obj.image_id,
            DETECTED_OBJECTS: [
               BoundedBoxObject(*(15, 15, 250, 225, 'pikachu', 0.5, ''))
            ],
            DETECTION_METHOD: BBOX
      }
      if self.enable_draw_bbox:
            image_dict[DRAWN_IMAGE_PATH] = str(
               Path(Path.cwd(), '%s_out.png' % image_obj.image_id))
            ImageHandler.draw_bbox(image_obj.pil_image_obj, image_dict[DETECTED_
↪OBJECTS])
            ImageHandler.save(image_obj.pil_image_obj, image_dict[DRAWN_IMAGE_PATH])

      detection_result = DetectionResult(image_dict)
      return detection_result
```

1.1.4 We can now run a fake example

always detect and draw bbox at same place

```
from eyewitness.result_handler.db_writer import BboxPeeweeDbWriter
from peewee import SqliteDatabase
import arrow
```

(continues on next page)

(continued from previous page)

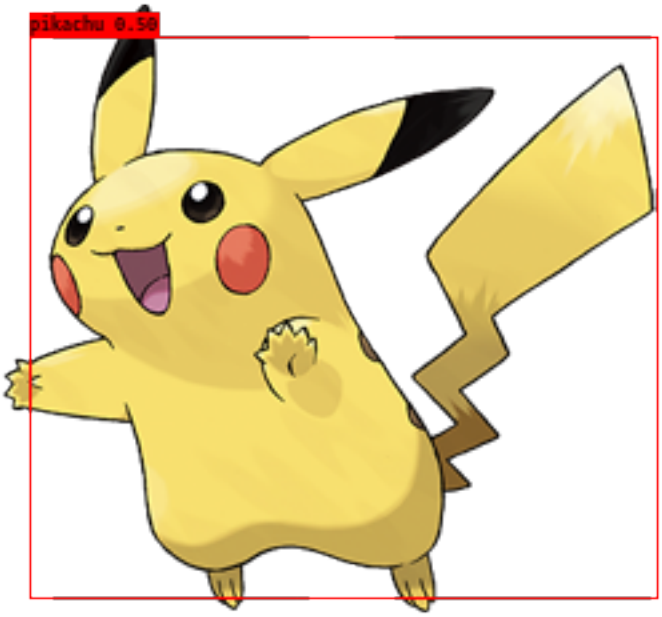
```
# init InMemoryImageProducer
image_producer = InMemoryImageProducer('pikachu.png')

# init FakePikachuDetector
object_detector = FakePikachuDetector()

# prepare detection result handler
database = SqliteDatabase("example.sqlite")
bbox_sqlite_handler = BboxPeeweeDbWriter(database)

for image_obj in image_producer.produce_image():
    # generate the image_obj
    bbox_sqlite_handler.register_image(image_obj.image_id, {})  # register the image_
↪info
    detection_result = object_detector.detect(image_obj)
    bbox_sqlite_handler.handle(detection_result)    # insert detection bbox result
```

which will keeping generate pikachu image, and write detection result into db



1.1.5 Real Detector Implement with Yolov3

start with the yolov3 Implement

the repo implements:

- naive_detector.py: wrapper the detector
- eyewitness_evaluation.py: run a evaluation example
- end2end_detector.py: a end2end detector example with webcam
- detector_with_flask.py: a end2end detector example with flask server

a naive detector example

```
class YoloV3DetectorWrapper(ObjectDetector):
   def __init__(self, model_config, threshold=0.5):
      self.core_model = YOLO(**vars(model_config))
      self.threshold = threshold

   def detect(self, image_obj) -> DetectionResult:
      (out_boxes, out_scores, out_classes) = self.core_model.predict(image_obj.pil_
↪image_obj)
      detected_objects = []
      for bbox, score, label_class in zip(out_boxes, out_scores, out_classes):
            label = self.core_model.class_names[label_class]
            y1, x1, y2, x2 = bbox
            if score > self.threshold:
               detected_objects.append(BoundedBoxObject(x1, y1, x2, y2, label, score,
↪''))

      image_dict = {
            'image_id': image_obj.image_id,
            'detected_objects': detected_objects,
      }
      detection_result = DetectionResult(image_dict)
      return detection_result
```

also there is a docker example in the docker/yolov3_pytorch

1.1.6 Docker examples

more with real detector examples with docker here

1.2 ImageId

a module used to represent a image, and store image information

class eyewitness.image_id.**ImageId**(*channel*, *timestamp*, *file_format='jpg'*)

Bases: object

ImageId is used to standardize the image_id format

Parameters

- **channel** (*str*) – channel of image comes
- **timestamp** (*int*) – timestamp of image arrive time
- **format** (*str*) – type of image

classmethod from_str(*image_id_str*)

serialize image_id from string, the seperator of image is double dash –

Parameters **image_id_str** (*str*) – a string with pattern {chanel}–{timestamp}–{fileformat}

e.g: "channel–12345567–jpg" (separated by a double dash)

Returns **image_id** – a ImageId obj

Return type *ImageId*

class eyewitness.image_id.**ImageRegister**
Bases: object

insert_image_info(*raw_image_path*)
abstract method which need to be implement: how to insert/record image information

Parameters

- **image_id** (ImageId) – ImageId obj
- **raw_image_path** (*str*) – the path of raw image

register_image(*image_id*, *meta_dict*)
interface for ImageRegister to register_image

1.3 Image Utils

Util methods for operation on image

class eyewitness.image_utils.**Image**(*image_id*, *raw_image_path=None*, *pil_image_obj=None*)
Bases: object

Image object is use to represent a Image in whole eyewitness project

To initialize a Image obj, image_id is required, and one of raw_image_path, pil_image_obj should be given, while only giving raw_image_path is kind of lazy evaluation, will read the image only when image_obj.pil_image_obj called

Parameters

- **image_id** (ImageId) – the id of image
- **raw_image_path** (*Optional[str]*) – the raw image path
- **pil_image_obj** (*Optional[PIL.Image.Image]*) – the pil image obj

fetch_bbox_pil_objs(*bbox_objs*)

Parameters

- **bbox_objs** (*List[BoundedBoxObject]*) – List of bbox objs, which used to generate bbox pil_image_obj
- **Returns** –
- **--------** –
- **output_list** (*List[PIL.Image.Image]*) –

pil_image_obj
pil_image_obj is a property of the Image, if _pil_image_obj exist will directly return the obj, else will read from raw_image_path.

class eyewitness.image_utils.**ImageHandler**
Bases: object

util functions for image processing including: save, read from file, read from bytes, draw bounding box.

classmethod draw_bbox(*image*, *detections*, *colors=None*, *font_path='/home/docs/checkouts/readthedocs.org/user_builds/*
Medium.otf')
draw bbox on to image.

Parameters

- **image** (*PIL.Image.Image*) – image to be draw
- **detections** (*List[BoundedBoxObject]*) – bbox to draw
- **colors** (*Optional[dict]*) – color to be used
- **font_path** (*str*) – font to be used

classmethod read_image_bytes(*image_byte*)
PIL.Image.open support BytesIO input.

Parameters **image_path** (*BytesIO*) – read image from ByesIO obj

Returns **pil_image_obj** – PIL.Image.Image instance

Return type PIL.Image.Image

classmethod read_image_file(*image_path*)
PIL.Image.open read from file.

Parameters **image_path** (*str*) – source image path

Returns **pil_image_obj** – PIL.Image.Image instance

Return type PIL.Image.Image

classmethod save(*image*, *output_path*)

Parameters

- **image** (*PIL.Image*) – image obj
- **output_path** (*str*) – path to be save

class eyewitness.image_utils.**ImageProducer**
Bases: object

ImageProducer abstract class, should produce_method property and produce_image function

produce_image()

produce_method

class eyewitness.image_utils.**PostBytesImageProducer**(*host*, *protocol='http'*)
Bases: *eyewitness.image_utils.ImageProducer*

PostBytes Image Producer, will sent the image bytes to destination by Http post

produce_image(*image_id*, *image_bytes*, *raw_image_path=None*)

produce_method

class eyewitness.image_utils.**PostFilePathImageProducer**(*host*, *protocol='http'*)
Bases: *eyewitness.image_utils.ImageProducer*

PostFilePath Image Producer, will sent the image_path string to destination by Http post

produce_image(*image_id*, *raw_image_path*)

produce_method

eyewitness.image_utils.**resize_and_stack_image_objs**(*resize_shape*, *pil_image_objs*)
resize images and concat into numpy array

Parameters

- **resize_shape** (*tuple[int]*) – the target resize shape (w, h)
- **pil_image_objs** (*List[PIL.Image.Image]*) – List of image objs

Returns **batch_images_array**

Return type np.array with shape (n, w, h, c)

eyewitness.image_utils.**swap_channel_rgb_bgr**(*image*)
reverse the color channel image: convert image (w, h, c) with channel rgb -> bgr, bgr -> rgb.

Parameters **image** (*np.array*) –

Returns **image**

Return type np.array

1.4 Object Detector

a module define the object detector interface

class eyewitness.object_detector.**ObjectDetector**
Bases: object

Abstract class used to wrapper object detector

detect(*image_obj*)
[abstract method] need to implement detection method which return DetectionResult obj

Parameters **image_obj** (*eyewitness.image_util.Image*) –

Returns **DetectionResult** – the detected result of given image

Return type *DetectionResult*

detection_method
detection_method for the ObjectDetector is BBOX

Returns **detection_method**

Return type String

valid_labels
[abstract property] the valid_labels of this detecotr e.g. set(['person', 'pikachu' ...]) this will be used while want to evaluation the detector

Returns **valid_labels**

Return type set[String]

1.5 Detection Utils

Utils modules used for detection

class eyewitness.detection_utils.**DetectionResult**(*image_dict*)
Bases: object

represent detection result of a image.

Parameters **image_dict** (*dict*) –

- detection_method: detection_method str
- detected_objects: List[tuple], list of detected obj (optional)
- drawn_image_path: str, path of drawn image (optional)

- image_id: image_id obj

detected_objects
List of detected objects in the image

Type List[object]

drawn_image_path
drawn_image_path

Type str

classmethod from_json(*json_str*)

image_id
image_id obj

Type *ImageId*

to_json_dict()

Returns **image_dict** – the dict repsentation of detection_result

Return type dict

class eyewitness.detection_utils.**DetectionResultHandler**
Bases: object

a abstract class design to handle detection result need to implement:

- function: _handle(self, detection_result)
- property: detection_method

detection_method

handle(*detection_result*)
wrapper of _handle function with the check of detection_method with detection_result.

Parameters **detection_result** (DetectionResult) –

1.6 Detection Result Filter

Utils modules used for filter detection result

class eyewitness.detection_result_filter.**DetectionResultFilter**
Bases: object

apply(*detection_result*)

detection_method

class eyewitness.detection_result_filter.**FeedbackBboxDeNoiseFilter**(*database*, *decay=0.9*, *iou_threshold=0.7*, *collect_feedback_period=172800*, *detection_threshold=0.5*, *time_check_period=None*)
Bases: *eyewitness.detection_result_filter.DetectionResultFilter*

a Bbox DeNoise filter, which will read false alert bbox from tables: FalseAlertFeedback, BboxDetectionResult, and apply filter onto the detection result

check_proxy_db()
check if the db proxy is correct one, if not initialize again.

detection_method
String BBOX

Type detection_method

update_false_alert_feedback_bbox()
collect_bbox_false_alert_information

1.7 Evaluation Utils

used to calculate the detector performance currently support mAP for object detection

class eyewitness.evaluation.**BboxMAPEvaluator**(*iou_threshold=0.5*, *dataset_mode='TEST_ONLY'*, *logging_frequency=100*)

Bases: *eyewitness.evaluation.Evaluator*

evaluate the bbox mAP score

static calculate_average_precision(*recall*, *precision*)

calculate_label_ap(*valid_labels*, *detected_objs*, *gt_objs*, *gt_label_count*)
refactor the evaluation from https://github.com/rafaelpadilla/Object-Detection-Metrics

evaluation_method

class eyewitness.evaluation.**Evaluator**

Bases: object

evaluate(*detector*, *dataset*)

evaluation_method

1.8 A BboxMAPEvaluator Example

```
# a evaluation example with yolov3 detector
# https://github.com/penolove/keras-yolo3/blob/eyeWitnessWrapper/eyewitness_
↪evaluation.py
from eyewitness.config import DATASET_TEST_ONLY
dataset_folder = 'VOC2007'
dataset_VOC_2007 = BboxDataSet(dataset_folder, 'VOC2007')
object_detector = YoloV3DetectorWrapper(args, threshold=0.0)
bbox_map_evaluator = BboxMAPEvaluator(dataset_mode=DATASET_TEST_ONLY)
# which will lead to ~0.73
print(bbox_map_evaluator.evaluate(object_detector, dataset_VOC_2007)['mAP'])
```

1.9 Audience Id

a module used to represent a audience, and store audience information

class eyewitness.audience_id.**AudienceId**(*platform_id*, *user_id*)
Bases: object

the target of AudienceId is used to standardize the AudienceId format,

Parameters

- **platform_id** (*str*) – platform_id of feedback user from
- **user_id** (*str*) – id of feedback user

classmethod from_str(*audience_id_str*)

Parameters **audience_id_str** (*str*) – a string with pattern {platform}–{audience_id} e.g: "line–minhan_hgdfmjg2715".

Returns **audience_id** – a AudienceId obj

Return type *AudienceId*

class eyewitness.audience_id.**AudienceRegister**
Bases: object

Abstract Class for handling audience registration

insert_registered_user(*audience_id*, *register_time*, *description*)
abstract method for register audience id

register_audience(*audience_id*, *meta_dict*)
register audience

Parameters

- **audience_id** (AudienceId) – audience information
- **meta_dict** (*dict*) – additional information

1.10 Feedback Msg Utils

Utils modules used for Feedback Msg

class eyewitness.feedback_msg_utils.**FeedbackMsg**(*feedback_dict*)
Bases: object

represent the Feedback msg

Parameters **feedback_dict** (*dict*) –

- **audience_id: AudienceId** the audience who feedback the msg
- **feedback_method: str** which kind of feedback
- **image_id: ImageId** the ImageId related to feedback
- **feedback_meta: str** misc feedback msg
- **feedback_msg_objs: List[tuple]** feedback objs (e.g. bboxs)
- **receive_time: int** the timestamp receive the msg

audience_id
AudienceId obj

Type *AudienceId*

feedback_meta
feedback_meta str

Type str

feedback_msg_objs
List of msg named tuple objs

Type List[tuples]

classmethod from_json(*json_str*)

Parameters **json_str** (*str*) – feedback_msg json str

Returns **feedback_msg_obj** – a feedback msg instance

Return type *FeedbackMsg*

image_id
ImageId obj

Type *ImageId*

is_false_alert
is false_alert or not

Type bool

receive_time
received timestamp

Type int

to_json_dict()

Returns **image_dict** – the dict repsentation of detection_result

Return type dict

class eyewitness.feedback_msg_utils.**FeedbackMsgHandler**
Bases: object

Abstract class for FeedbackMsgHandler

with a abstract method *_handle(feedback_msg)* used to handle feedback msg

feedback_method
feedback_method

Type str

handle(*feedback_msg*)
a wrapper for *_handle(feedback_msg)* and *feedback_method* check

1.11 Handler Example

Handlers for detection results

1.11.1 DB Writer

class eyewitness.result_handler.db_writer.**BboxNativeSQLiteDbWriter**(*db_path*)
Bases: *eyewitness.detection_utils.DetectionResultHandler*, *eyewitness.image_id.ImageRegister*

Parameters **db_path** (*str*) – database path

create_db_table()
create ImageInfo, BboxDetectionResult table if table not exist

detection_method
BBOX

Type str

insert_detection_objs(*image_id*, *detected_objects*)
insert detection results into db.

Parameters

- **image_id** (*str*) – image_id
- **detected_objects** (*List[BoundedBoxObject]*) – detected objects

insert_image_info(*image_id*, *raw_image_path=None*)
insert image_info which used for unit-test

Parameters

- **image_id** (*str*) – image_id
- **raw_image_path** (*str*) – the path of raw image stored

update_image_drawn_image_path(*image_id*, *drawn_image_path*)
update db image_id.drawn_image_path

class eyewitness.result_handler.db_writer.**BboxPeeweeDbWriter**(*database*, *auto_image_registration=False*)
Bases: *eyewitness.detection_utils.DetectionResultHandler*, *eyewitness.image_id.ImageRegister*

Parameters

- **database** (*peewee.Database*) – peewee db obj
- **auto_image_registration** (*Bool*) – enable the auto_image_registration will check if image registered or not which might make the handle function more slowly

check_proxy_db()
check if the db proxy is correct one, if not initialize again.

create_db_table()
create ImageInfo, BboxDetectionResult table if table not exist

detection_method

insert_detection_objs(*image_id*, *detected_objects*)
insert detection results into db.

Parameters

- **image_id** (*str*) – image_id
- **detected_objects** (*List[BoundedBoxObject]*) – detected objects

insert_image_info(*image_id*, *raw_image_path=None*)
insert image_info which used for unit-test

Parameters

- **image_id** (*ImageId obj*) – image_id obj (including channel, timestamp, file-format)
- **raw_image_path** (*str*) – the path of raw image stored

update_image_drawn_image_path(*image_id*, *drawn_image_path*)
update db image_id.drawn_image_path

class eyewitness.result_handler.db_writer.**FalseAlertPeeweeDbWriter**(*database*)
Bases: *eyewitness.feedback_msg_utils.FeedbackMsgHandler*, *eyewitness.audience_id.AudienceRegister*, *eyewitness.image_id.ImageRegister*

Parameters **database** (*peewee.Database*) – peewee db obj

check_proxy_db()
check if the db proxy is correct one, if not initialize again.

create_db_table()
create ImageInfo, RegisteredAudience, FalseAlertFeedback table if table not exist

feedback_method
feedback_method

Type str

insert_feedback_obj(*feedback_msg*)
insert feedback obj into db.

Parameters **feedback_msg** (FeedbackMsg) –

insert_image_info(*image_id*, *raw_image_path=None*)
insert image_info which used for unit-test

Parameters

- **image_id** (*str*) – image_id
- **raw_image_path** (*str*) – the path of raw image stored

insert_registered_user(*audience_id*, *register_time*, *description*)
insert image_info which used for unit-test

Parameters

- **audience_id** (AudienceId) –
- **register_time** (*int*) –
- **description** (*str*) –

1.12 ORM DB Models

orm models for Eyewitness with the support of peewee

1.12.1 ImageInfo

```
class ImageInfo(BaseModel):
    image_id = CharField(unique=True, primary_key=True)
    channel = CharField()
    file_format = CharField()
    timestamp = TimestampField()
    raw_image_path = CharField(null=True)
    drawn_image_path = CharField(null=True)
```

1.12.2 BboxDetectionResult

```
class BboxDetectionResult(BaseModel):
    image_id = ForeignKeyField(ImageInfo)
    x1 = IntegerField()
    x2 = IntegerField()
    y1 = IntegerField()
    y2 = IntegerField()
    label = CharField()
    meta = CharField()
    score = DoubleField()
```

1.12.3 RegisteredAudience

```
class RegisteredAudience(BaseModel):
    audience_id = CharField(unique=True, primary_key=True)
    user_id = CharField(null=False)
    platform_id = CharField(null=False)
    register_time = TimestampField()
    description = CharField()
```

1.12.4 FalseAlertFeedback

```
class FalseAlertFeedback(BaseModel):
    # peewee didn't support compositeKey as foreignKey, using field to specify field
    audience_id = ForeignKeyField(RegisteredAudience)
    image_id = ForeignKeyField(ImageInfo, null=True)
    receive_time = TimestampField()
    feedback_meta = CharField()
    # TODO: if the is_false_alert field needed??
    is_false_alert = BooleanField()
```

1.12.5 BboxAnnotationFeedback

```
class BboxAnnotationFeedback(BaseModel):
    # peewee didn't support compositeKey as foreignKey, using field to specify field
    audience_id = ForeignKeyField(RegisteredAudience)
    image_id = ForeignKeyField(ImageInfo, null=True)
    receive_time = TimestampField()
```

(continues on next page)

(continued from previous page)

```
    feedback_meta = CharField()
    is_false_alert = BooleanField()
    x1 = IntegerField()
    x2 = IntegerField()
    y1 = IntegerField()
    y2 = IntegerField()
    label = CharField()
```

1.13 DataSet Utils

used to export the detected data which can used for retrain/fine tune the model

class eyewitness.dataset_util.**BboxDataSet**(*dataset_folder*, *dataset_name*, *valid_labels=None*)

Bases: object

generate DataSet with same format as VOC object detections:

<dataset_folder>/Annotations/<image_name>.xml

<dataset_folder>/JPEGImages/<image_name>.jpg

<dataset_folder>/ImageSets/Main/trainval.txt

<dataset_folder>/ImageSets/Main/test.txt

convert_into_darknet_format()

dataset_iterator(*with_gt_objs=True*, *mode='TEST_ONLY'*)

dataset_type

generate_train_test_list(*overwrite=True*, *train_ratio=0.9*)

generate train and test list

Parameters

- **overwrite** (*bool*) – if overwrite and file not exit will regenerate the train, test list
- **train_ratio** (*float*) – the ratio used to sample train, test list, should between 0~1

get_selected_images(*mode='TEST_ONLY'*)

get_valid_labels()

ground_truth_iterator(*selected_images*)

ground_truth interator

Parameters **mode** (*str*) – the mode to iterate the dataset

Returns **gt_object_generator** – ground_truth_object generator, with first item if the ImageId

Return type Generator[(*ImageId*, List[BoundedBoxObject])]

image_obj_iterator(*selected_images*)

generate eyewitness Image obj from dataset

Parameters **mode** (*str*) – the mode to iterate the dataset

Returns **image_obj_generator** – eyewitness Image obj generator

Return type Generator[*eyewitness.image_utils.Image*]

store_and_convert_darknet_bbox_tuples(*dataset_file*, *selected_images*, *images_dir*, *labels_dir*, *label2idx*, *logging_frequency=100*)

testing_set

training_and_validation_set

classmethod union_bbox_datasets(*datasets*, *output_dataset_folder*, *dataset_name*, *filter_labels=None*, *remove_empty_labels_file=False*)
union bbox datasets and copy files to the given output_dataset

valid_labels
the valid_labels in the dataset

eyewitness.dataset_util.**add_filename_prefix**(*filename*, *prefix*)

eyewitness.dataset_util.**copy_image_to_output_dataset**(*filename*, *src_dataset*, *jpg_images_folder*, *anno_folder*, *file_fp*, *filter_labels=None*, *remove_empty_labels_file=False*)
move annotation, jpg file from src_dataset to file destination, add prefix to filename and print to id list file

Parameters

- **filename** (*str*) – ori filename
- **src_dataset** (BboxDataSet) – source dataset
- **jpg_images_folder** (*str*) – destination jpg file folder
- **anno_folder** (*str*) – destination annotation file folder
- **file_fp** – the file pointer used to export the id list
- **filter_labels** (*Optional[set[String]]*) – used for filtering label for the destination dataset

eyewitness.dataset_util.**create_bbox_dataset_from_eyewitness**(*database*, *valid_classes*, *output_dataset_folder*, *dataset_name*)
generate bbox dataset from eyewitness requires:

- FalseAlertFeedback table: remove images with false-alert feedback
- BboxDetectionResult: get images with selected classes objects

eyewitness.dataset_util.**generate_etree_obj**(*image_id*, *detected_objects*, *dataset_name*)

Parameters

- **image_id** (*str*) – image_id as filename
- **detected_objects** – detected_objects obj from detected_objects table
- **dataset_name** (*str*) – dataset_name

eyewitness.dataset_util.**parse_xml_obj**(*obj*)

eyewitness.dataset_util.**read_ori_anno_and_store_filered_result**(*ori_anno_file*, *dest_anno_file*, *filter_labels*, *remove_empty_labels_file*)
read the original annotation file, filter objects with valid labels export to the dest_anno_file

Parameters

- **ori_anno_file** (*str*) – original annotation file
- **dest_anno_file** (*str*) – destination annotation file
- **filter_labels** (*Optional[set[String]]*) – filter the labels
- **remove_empty_labels_file** (*bool*) – remove the image if it don't have obj

1.14 MOT Module

Modules related to MOT(multiple object tracking)

1.14.1 Video abstract

class eyewitness.mot.video.**FilesAsVideoData**(*image_files*, *frame_shape=None*, *frame_rate=3*)

Bases: *eyewitness.mot.video.VideoData*

frame_rate

frame_shape

n_frames

to_video(*video_output_path*, *ffmpeg_quiet=True*)

class eyewitness.mot.video.**FolderAsVideoData**(*images_dir*, *file_template='*[0-9].jpg'*)

Bases: *eyewitness.mot.video.FilesAsVideoData*

class eyewitness.mot.video.**Mp4AsVideoData**(*video_file*, *ffmpeg_quiet=True*, *in_memory=True*)

Bases: *eyewitness.mot.video.VideoData*

frame_rate

frame_shape

n_frames

class eyewitness.mot.video.**VideoData**

Bases: object

this class were used to represent a Video (List of Frames)

frame_rate

frame_shape

n_frames

eyewitness.mot.video.**is_program_exists**(*program*)

since the python-ffmpeg needs the host install ffmpeg first thus we need a method that can used to find executable file exists

program: str the executable file to fine

is_file_exists: bool return the executable file exists or not

1.14.2 Tracker abstract

class eyewitness.mot.tracker.**ObjectTracker**
Bases: object

track(*video_data*)

Parameters **video_data** (VideoData) – the video data to be tracked

Returns **video_tracked_result** – the tracked video result

Return type *VideoTrackedObjects*

1.14.3 Evaluation

class eyewitness.mot.evaluation.**VideoTrackedObjects**
Bases: collections.defaultdict

actually a VideoTrackedObjects object is a subclass of defaultdict with list and expected result were Dict[int, List[BoundedBoxObject]]

classmethod from_dict(*tracked_obj_dict*)

classmethod from_tracked_file(*trajectory_file*, *ignore_gt_flag=False*)
parsed the trajectory file, and reuse the BoundedBoxObject class

Parameters **track_file** (*str*) – the file path of object tracking ground_truth, format is *<frame>, <id>, <bb_left>, <bb_top>, <bb_width>, <bb_height>, <conf>...*

Returns **parsed_tracked_objects** – key is the frame_idx, value is the objects detected in this frame and the label field in BoundedBoxObject were set as object_id

Return type Dict[int, List[BoundedBoxObject]]

to_file(*dest_file*)

eyewitness.mot.evaluation.**mot_evaluation**(*video_gt_objects*, *video_tracked_objects*, *threshold=0.5*)
with the help of motmetrics we can evaluate our mot tracker

video_gt_objects: Dict[int, List[BoundedBoxObject]] ground_truth object of video, key is the frame_idx, value is the objects detected in this frame and the label field in BoundedBoxObject were set as object_id

video_tracked_objects: Dict[int, List[BoundedBoxObject]] predicted mot result of video, key is the frame_idx, value is the objects detected in this frame and the label field in BoundedBoxObject were set as object_id

summary: Dataframe the dataframe of evaluation result with the fields used in MOT2019 https://motchallenge.net/results/CVPR_2019_Tracking_Challenge/

1.14.4 Visualize

eyewitness.mot.visualize_mot.**draw_tracking_result**(*parsed_tracked_objects*, *color_list*, *video_obj*, *output_images_dir=None*, *output_video_path=None*, *n_trajectory=50*, *ffmpeg_quiet=True*)
this method used to draw the tracked result back to original video notice that, if you want to export to output_video_path, you need to install it in your host, e.g. apt install ffmpeg

Parameters

- **parsed_tracked_objects** (*Dict[int, List[BoundedBoxObject]]*) – key is the frame_idx, value is the objects detected in this frame and the label field in BoundedBoxObject were set as object_id
- **color_list** (*List[tuple[int]]*) – the color_list used to draw each object_id
- **video_obj** (VideoData) – the original video object
- **output_images_dir** (*Optional[str]*) – the dir used to stored drawn image, the stored template is Path(output_images_dir, "%s.jpg" % str(t).zfill(6)), t is current frame number
- **output_video_path** (*Optional[str]*) – the output path of video
- **n_trajectory** (*int*) – the number of previous point to be drawn
- **ffmpeg_quiet** (*bool*) – route the ffmpeg_quiet logging to stdout or not

Python Module Index

e

eyewitness.audience_id, 12
eyewitness.dataset_util, 18
eyewitness.detection_result_filter, 11
eyewitness.detection_utils, 10
eyewitness.evaluation, 12
eyewitness.feedback_msg_utils, 13
eyewitness.image_id, 7
eyewitness.image_utils, 8
eyewitness.mot.evaluation, 21
eyewitness.mot.tracker, 21
eyewitness.mot.video, 20
eyewitness.mot.visualize_mot, 21
eyewitness.object_detector, 10
eyewitness.result_handler.db_writer, 15

Index

A

add_filename_prefix() (*in module eyewitness.dataset_util*), 19
apply() (*eyewitness.detection_result_filter.DetectionResultFilter method*), 11
audience_id (*eyewitness.feedback_msg_utils.FeedbackMsg attribute*), 13
AudienceId (*class in eyewitness.audience_id*), 12
AudienceRegister (*class in eyewitness.audience_id*), 13

B

BboxDataSet (*class in eyewitness.dataset_util*), 18
BboxMAPEvaluator (*class in eyewitness.evaluation*), 12
BboxNativeSQLiteDbWriter (*class in eyewitness.result_handler.db_writer*), 15
BboxPeeweeDbWriter (*class in eyewitness.result_handler.db_writer*), 15

C

calculate_average_precision() (*eyewitness.evaluation.BboxMAPEvaluator static method*), 12
calculate_label_ap() (*eyewitness.evaluation.BboxMAPEvaluator method*), 12
check_proxy_db() (*eyewitness.detection_result_filter.FeedbackBboxDeNoiseFilter method*), 12
check_proxy_db() (*eyewitness.result_handler.db_writer.BboxPeeweeDbWriter method*), 15
check_proxy_db() (*eyewitness.result_handler.db_writer.FalseAlertPeeweeDbWriter method*), 16
convert_into_darknet_format() (*eyewitness.dataset_util.BboxDataSet method*), 18
copy_image_to_output_dataset() (*in module eyewitness.dataset_util*), 19
create_bbox_dataset_from_eyewitness() (*in module eyewitness.dataset_util*), 19
create_db_table() (*eyewitness.result_handler.db_writer.BboxNativeSQLiteDbWriter method*), 15
create_db_table() (*eyewitness.result_handler.db_writer.BboxPeeweeDbWriter method*), 15
create_db_table() (*eyewitness.result_handler.db_writer.FalseAlertPeeweeDbWriter method*), 16

D

dataset_iterator() (*eyewitness.dataset_util.BboxDataSet method*), 18
dataset_type (*eyewitness.dataset_util.BboxDataSet attribute*), 18
detect() (*eyewitness.object_detector.ObjectDetector method*), 10
detected_objects (*eyewitness.detection_utils.DetectionResult attribute*), 11
detection_method (*eyewitness.detection_result_filter.DetectionResultFilter attribute*), 11
detection_method (*eyewitness.detection_result_filter.FeedbackBboxDeNoiseFilter attribute*), 12
detection_method (*eyewitness.detection_utils.DetectionResultHandler attribute*), 11
detection_method (*eyewitness.object_detector.ObjectDetector attribute*), 10
detection_method (*eyewitness.result_handler.db_writer.BboxNativeSQLiteDbWriter attribute*), 15

detection_method (*eyewitness.result_handler.db_writer.BboxPeeweeDbWriter attribute*), 15
DetectionResult (*class in eyewitness.detection_utils*), 10
DetectionResultFilter (*class in eyewitness.detection_result_filter*), 11
DetectionResultHandler (*class in eyewitness.detection_utils*), 11
draw_bbox() (*eyewitness.image_utils.ImageHandler class method*), 8
draw_tracking_result() (*in module eyewitness.mot.visualize_mot*), 21
drawn_image_path (*eyewitness.detection_utils.DetectionResult attribute*), 11

E

evaluate() (*eyewitness.evaluation.Evaluator method*), 12
evaluation_method (*eyewitness.evaluation.BboxMAPEvaluator attribute*), 12
evaluation_method (*eyewitness.evaluation.Evaluator attribute*), 12
Evaluator (*class in eyewitness.evaluation*), 12
eyewitness.audience_id (*module*), 12
eyewitness.dataset_util (*module*), 18
eyewitness.detection_result_filter (*module*), 11
eyewitness.detection_utils (*module*), 10
eyewitness.evaluation (*module*), 12
eyewitness.feedback_msg_utils (*module*), 13
eyewitness.image_id (*module*), 7
eyewitness.image_utils (*module*), 8
eyewitness.mot.evaluation (*module*), 21
eyewitness.mot.tracker (*module*), 21
eyewitness.mot.video (*module*), 20
eyewitness.mot.visualize_mot (*module*), 21
eyewitness.object_detector (*module*), 10
eyewitness.result_handler.db_writer (*module*), 15

F

FalseAlertPeeweeDbWriter (*class in eyewitness.result_handler.db_writer*), 16
feedback_meta (*eyewitness.feedback_msg_utils.FeedbackMsg attribute*), 13
feedback_method (*eyewitness.feedback_msg_utils.FeedbackMsgHandler attribute*), 14
feedback_method (*eyewitness.result_handler.db_writer.FalseAlertPeeweeDbWriter attribute*), 16
feedback_msg_objs (*eyewitness.feedback_msg_utils.FeedbackMsg attribute*), 14
FeedbackBboxDeNoiseFilter (*class in eyewitness.detection_result_filter*), 11
FeedbackMsg (*class in eyewitness.feedback_msg_utils*), 13
FeedbackMsgHandler (*class in eyewitness.feedback_msg_utils*), 14
fetch_bbox_pil_objs() (*eyewitness.image_utils.Image method*), 8
FilesAsVideoData (*class in eyewitness.mot.video*), 20
FolderAsVideoData (*class in eyewitness.mot.video*), 20
frame_rate (*eyewitness.mot.video.FilesAsVideoData attribute*), 20
frame_rate (*eyewitness.mot.video.Mp4AsVideoData attribute*), 20
frame_rate (*eyewitness.mot.video.VideoData attribute*), 20
frame_shape (*eyewitness.mot.video.FilesAsVideoData attribute*), 20
frame_shape (*eyewitness.mot.video.Mp4AsVideoData attribute*), 20
frame_shape (*eyewitness.mot.video.VideoData attribute*), 20
from_dict() (*eyewitness.mot.evaluation.VideoTrackedObjects class method*), 21
from_json() (*eyewitness.detection_utils.DetectionResult class method*), 11
from_json() (*eyewitness.feedback_msg_utils.FeedbackMsg class method*), 14
from_str() (*eyewitness.audience_id.AudienceId class method*), 13
from_str() (*eyewitness.image_id.ImageId class method*), 7
from_tracked_file() (*eyewitness.mot.evaluation.VideoTrackedObjects class method*), 21

G

generate_etree_obj() (*in module eyewitness.dataset_util*), 19
generate_train_test_list() (*eyewitness.dataset_util.BboxDataSet method*), 18
get_selected_images() (*eyewitness.dataset_util.BboxDataSet method*), 18

get_valid_labels() (*eyewitness.dataset_util.BboxDataSet method*), 18
ground_truth_iterator() (*eyewitness.dataset_util.BboxDataSet method*), 18

H

handle() (*eyewitness.detection_utils.DetectionResultHandler method*), 11
handle() (*eyewitness.feedback_msg_utils.FeedbackMsgHandler method*), 14

I

Image (*class in eyewitness.image_utils*), 8
image_id (*eyewitness.detection_utils.DetectionResult attribute*), 11
image_id (*eyewitness.feedback_msg_utils.FeedbackMsg attribute*), 14
image_obj_iterator() (*eyewitness.dataset_util.BboxDataSet method*), 18
ImageHandler (*class in eyewitness.image_utils*), 8
ImageId (*class in eyewitness.image_id*), 7
ImageProducer (*class in eyewitness.image_utils*), 9
ImageRegister (*class in eyewitness.image_id*), 7
insert_detection_objs() (*eyewitness.result_handler.db_writer.BboxNativeSQLiteDbWriter method*), 15
insert_detection_objs() (*eyewitness.result_handler.db_writer.BboxPeeweeDbWriter method*), 15
insert_feedback_obj() (*eyewitness.result_handler.db_writer.FalseAlertPeeweeDbWriter method*), 16
insert_image_info() (*eyewitness.image_id.ImageRegister method*), 8
insert_image_info() (*eyewitness.result_handler.db_writer.BboxNativeSQLiteDbWriter method*), 15
insert_image_info() (*eyewitness.result_handler.db_writer.BboxPeeweeDbWriter method*), 15
insert_image_info() (*eyewitness.result_handler.db_writer.FalseAlertPeeweeDbWriter method*), 16
insert_registered_user() (*eyewitness.audience_id.AudienceRegister method*), 13
insert_registered_user() (*eyewitness.result_handler.db_writer.FalseAlertPeeweeDbWriter method*), 16
is_false_alert (*eyewitness.feedback_msg_utils.FeedbackMsg attribute*), 14
is_program_exists() (*in module eyewitness.mot.video*), 20

M

mot_evaluation() (*in module eyewitness.mot.evaluation*), 21
Mp4AsVideoData (*class in eyewitness.mot.video*), 20

N

n_frames (*eyewitness.mot.video.FilesAsVideoData attribute*), 20
n_frames (*eyewitness.mot.video.Mp4AsVideoData attribute*), 20
n_frames (*eyewitness.mot.video.VideoData attribute*), 20

O

ObjectDetector (*class in eyewitness.object_detector*), 10
ObjectTracker (*class in eyewitness.mot.tracker*), 21

P

parse_xml_obj() (*in module eyewitness.dataset_util*), 19
pil_image_obj (*eyewitness.image_utils.Image attribute*), 8
PostBytesImageProducer (*class in eyewitness.image_utils*), 9
PostFilePathImageProducer (*class in eyewitness.image_utils*), 9
produce_image() (*eyewitness.image_utils.ImageProducer method*), 9
produce_image() (*eyewitness.image_utils.PostBytesImageProducer method*), 9
produce_image() (*eyewitness.image_utils.PostFilePathImageProducer method*), 9
produce_method (*eyewitness.image_utils.ImageProducer attribute*), 9
produce_method (*eyewitness.image_utils.PostBytesImageProducer attribute*), 9
produce_method (*eyewitness.image_utils.PostFilePathImageProducer attribute*), 9

R

read_image_bytes() (*eyewitness.image_utils.ImageHandler class method*), 9

read_image_file() (*eyewitness.image_utils.ImageHandler class method*), 9
read_ori_anno_and_store_filered_result() (*in module eyewitness.dataset_util*), 19
receive_time (*eyewitness.feedback_msg_utils.FeedbackMsg attribute*), 14
register_audience() (*eyewitness.audience_id.AudienceRegister method*), 13
register_image() (*eyewitness.image_id.ImageRegister method*), 8
resize_and_stack_image_objs() (*in module eyewitness.image_utils*), 9

S

save() (*eyewitness.image_utils.ImageHandler class method*), 9
store_and_convert_darknet_bbox_tuples() (*eyewitness.dataset_util.BboxDataSet method*), 18
swap_channel_rgb_bgr() (*in module eyewitness.image_utils*), 10

T

testing_set (*eyewitness.dataset_util.BboxDataSet attribute*), 19
to_file() (*eyewitness.mot.evaluation.VideoTrackedObjects method*), 21
to_json_dict() (*eyewitness.detection_utils.DetectionResult method*), 11
to_json_dict() (*eyewitness.feedback_msg_utils.FeedbackMsg method*), 14
to_video() (*eyewitness.mot.video.FilesAsVideoData method*), 20
track() (*eyewitness.mot.tracker.ObjectTracker method*), 21
training_and_validation_set (*eyewitness.dataset_util.BboxDataSet attribute*), 19

U

union_bbox_datasets() (*eyewitness.dataset_util.BboxDataSet class method*), 19
update_false_alert_feedback_bbox() (*eyewitness.detection_result_filter.FeedbackBboxDeNoiseFilter method*), 12
update_image_drawn_image_path() (*eyewitness.result_handler.db_writer.BboxNativeSQLiteDbWriter method*), 15
update_image_drawn_image_path() (*eyewitness.result_handler.db_writer.BboxPeeweeDbWriter method*), 16

V

valid_labels (*eyewitness.dataset_util.BboxDataSet attribute*), 19
valid_labels (*eyewitness.object_detector.ObjectDetector attribute*), 10
VideoData (*class in eyewitness.mot.video*), 20
VideoTrackedObjects (*class in eyewitness.mot.evaluation*), 21