**V0.4**

**2016.3.12**

**storm\_url= 'http://drpc1v.cldisk2.tjhc.qihoo.net:3775/drpc/hdp-iri\_api'**

**现在用的是小水滴的测试库，只有一个线程处理，所以为了区分**

Qid应该自己设置一个，例如qsafe\_test

1. **请求方法**

storm 请求样例

1、请求json：

{

"version":"",

"method": "",

"data":{

"qid":"",

}

}

由于storm上复用同一接口，所以调用函数以参数的形式传入

qid是用户id。

2、返回json：

{

"rc":0,//请求正常

"msg":"",

data: {},

}

3、返回值含义（部分）

0:"ok",

1:"method not found",

2:"input invalid",

#meta class

101:" user\_id max limit",

102:"qid error",

103:"del fault",

#face\_verify

201:"no face",

202:"no face to tag",

203:" user\_id not exist,need create",

204:"face not exist",

205:"cluster center update fault",

206:"gound truth update fault",

**二、具体接口及样例**

**第一步:人脸检测和验证**

       输入参数除了图片数据，还有imageId，其中清晰正面半侧脸带有预测的UserID（或未知0）和faceId以及相似度(考虑到一张图片多张人脸的情况)，如果非清晰脸，全部未知(比如-1)（同时也不用来做分堆)。

Method: get\_face\_info

Param: qid imagedata（base64编码）image\_id

Return: rc data{[{ 'image\_id' 'face\_id' 'detection' ' p\_user\_id '(0未知 -1质量差) degree }] }

Example:

Request:

request = {

"version":"",

"method": " get\_face\_info ",

"data":{

" qid ": ‘qsafe\_test’,

"imagedata":encoded\_string,

" image\_id ": image\_id,

}

}

print json.dumps(request)

Return:

{

"rc":0,//请求正常

"msg":"",

data:{face\_infos:[ {

"user\_id" : 0,

"detection" : [862, 405, 58, 57],

"image\_id" : "24979971-36020144175-3-1448934419888.jpg",

"create\_time" : new Date("23/2/2016 22:44:16"),

"qid" : "003",

"degree":80,

"face\_id" : "24979971-36020144175-3-1448934419888.jpg\_0"

}]},

}

**第二步:分堆（聚类）**

1. 触发分堆:我们会输入qid列表(其中有些qid可能没有图片)和阀值(例如不小于100张才聚类)。分堆的结果需要人脸识别服务器保存。

Method: face\_pre\_cluster

Param: qids (qid list) face\_num\_thd

Return: rc

Example:

Request:

request = {

"version":"",

"method": " face\_pre\_cluster ",

"data":{

"qids":["003","004"],

"face\_num\_thd":23,

}

}

print json.dumps(request)

Return:

{

"rc":0,//请求正常

"msg":"",

data: {},

}

   2，获取分堆列表:人脸识别服务器需要告诉我们某一qid下未被分配UserID标识的分堆信息，包括每个堆的ID,以及此堆下的所有imageId及对应的faceId。

Method: get\_face\_cluster

Param: qid

Return: rc "cluster":[{"user\_id": 0, "faces":[{}]}]

Faces里面为faceinfos(list)：[{ 'image\_id' 'face\_id' 'detection' }]

["user\_id": 可选，目前不加，含义：本堆最可能的user\_id]

[user\_id是不存在的，下面sanple里面返回结果代表两个堆,外层list有两个dict元素,代表两个cluster]

Example:

Request:

request = {

"version":"",

"method": " get\_face\_cluster ",

"data":{

"qid":"003"

}

}

print json.dumps(request)

Return:

{

"rc":0,//请求正常

"msg":"",

data: {

"cluster": [{"faces": [{"detection": [630, 446, 75, 75], "image\_id": "24979971-36020144175-3-1448934419888.jpg", "face\_id": "24979971-36020144175-3-1448934419888.jpg\_1"}]}，{ "faces": [{"detection": [630, 446, 75, 75], "image\_id": "24979971-36020144175-3-1448934419888.jpg", "face\_id": "24979971-36020144175-3-1448934419888.jpg\_1"}]}]

},

}

**第三****步:打标签**

1，给某一分堆下选中图片打标签，然后我们输入分堆的ID,每张图片的faceId。如果此标签没有对应的UserID，需要新增加UserId。(增加见第四步)

   2，正反馈:我们提供用户选中的图片的faceId及对应的UserID。

   3，负反馈:我们提供用户选中的图片的faceId和来源(图片识别或者用户划分)及对应的UserID。

(1)打标签及正反馈

Method: tag\_face

Param: qid face\_ids(list) (image\_id) user\_id

Return: rc

Example:

Request:

request = {

"version":"",

"method": " tag\_face ",

"data":{

"qid":"003",

"user\_id":2,

" face\_ids":[" 24979971-36020144175-3-1448934419888.jpg\_0 "," 24979971-36020144175-3-1448934419888.jpg\_1"],

}

}

print json.dumps(request)

Return:

{

"rc":0,//请求正常

"msg":"",

data:{},

}

(2)负反馈

Method: de\_tag\_face

Param: qid face\_ids(list) (image\_id) user\_id

Return: rc

Example:

Request:

request = {

"version":"",

"method": " de\_tag\_face ",

"data":{

"qid":"003",

"user\_id":2,

" face\_ids":[" 24979971-36020144175-3-1448934419888.jpg\_0 "," 24979971-36020144175-3-1448934419888.jpg\_1"],

}

}

print json.dumps(request)

Return:

{

"rc":0,//请求正常

"msg":"",

data:{},

}

**第四步:增删用户**UserID

1. 增加

Method: meta\_add\_user

Param: qid

Return: rc user\_id

Example:

Request:

request = {

"version":"",

"method": "meta\_add\_user",

"data":{

"qid":"003",

}

}

print json.dumps(request)

Return:

{

"rc":0,//请求正常

"msg":"",

data: {

"user\_id":2,

},

}

["user\_id"变为数字]

1. 删除

提供用户选中UserID。

Method: meta\_del\_user

Param: qid user\_id

Return: rc

Example:

Request:

request = {

"version":"",

"method": "meta\_del\_user",

"data":{

"qid":"003",

"user\_id":2

}

}

print json.dumps(request)

Return:

{

"rc":0,//请求正常

"msg":"",

data:{},

}