利用Python恢复微信缩略图

2018-12-20 14:12

2.7k 字 12 分钟

最近手机空间不够，发现微信占了快12G的存储。觉得应该清理一下了，刷了一下知乎，看到不少人推荐用手机管家清理。想着既然是腾讯出的，有官方背书，应该值得一信。

遂下载安装，打开扫描一下，提示有快1G的缩略图可以清理。并且上面还写着，不影响使用。

点了一下，还真的清理了。但是，点开聊天记录，里面居然都是一片灰色的？？？缩略图全不见了，而且也不会重新生成。那这样要你何用？查找聊天记录根本不知道啥时候发了什么图。

不过倒也是符合微信的设计，让人摸不到头脑，无论是朋友圈的分组逻辑还是会话删除逻辑，还有文件超时逻辑，全都是设计的一头雾水。

强迫症不能忍，只能靠自己来捞回缩略图了。

分析

首先，根据判断，缩略图这些肯定不会存在服务器上。连图片一段时间没打开就会超时无法下载，缩略图这些无关紧要的东西，肯定不会占用服务器的空间。所以缩略图应该都是本地生成存储的。

在Android手机上，微信图片、音频等资源文件都存储在 `存储卡/tencent/MicroMsg` 这个目录下。

找到这个目录下那一串用户id，打开一看，N多层级的文件夹，翻了几个都没找到头绪。

只能尝试从聊天数据库中分析，找到一个导出微信数据库项目，把微信数据库导出来。使用SQLiteBrowser打开。

关于微信数据库的格式，网上一堆资料，就不再此重复了。

找到记录发送图片的消息，格式为

```
<msg>
    <img aeskey="xxxxxxx" encryver="1" cdnthumbaeskey="xxxxxxx" cdnthumburl="xxxxxxx"
cdnthumblength="3863" cdnthumbheight="150" cdnthumbwidth="88" cdnmidheight="0" cdnmidwidth="0"
cdnhdheight="0" cdnhdwidth="0" cdnmidimgurl="xxxxxxx" length="44965" cdnbigimgurl="xxxxxxx"
hdlength="40298" md5="xxxxxxx" />
</msg>
```

乍一看好多字段，但是大部分都是cdn相关的。发现一个字段 `md5` ，直觉表明，微信应该是通过md5来找到对应的图片。

做了一个实验，利用腾讯手机管家导出一张微信缩略图片，保存后的名称为 `th_390137f65d945a8886c9836249038470` ，

去掉开头的 `th_` ，剩余的长度和 `md5` 字段长度一样，而且格式也很像。

那么会不会微信的缩略图名称就是 `th_原图的md5` 呢？经过一番Google，在某个博客上看到作者提到，微信图片的存储路径为 `存储卡/tencent/MicroMsg/用户ID/image2/xx` 下。 `xx` 即为图片MD5开头两位字符。

马上用adb连上手机，根据导出来的微信聊天数据库，随便找了一张图片MD5，果然在对应文件夹下找到文件。pull到电脑上。用二进制方式打开，果然是 `ffd8` 开头， `ffd9` 结束,一张jpeg格式的图片。

并且在同一个文件夹下，也找到一个名为 `th_原图MD5` 的文件，push到电脑上，改后缀。Bingo！就是缩略图。

那么现在就清晰了，微信Android端的缩略图存储逻辑为：

`原图片MD5加上th_开头，并且去掉后缀名，保存在跟原图同一层级的文件夹下。`

在加载聊天记录时，微信应该会根据图片的MD5，直接拼成缩略图的路径，去加载缩略图。如果能找到文件则加载，否则就显示灰色方块。那么我们要做的工作就是，通过某种方式，把对应图片的缩略图生成后放到指定目录下。这样就能在聊天记录中看到缩略图了。

开工

思考了一下，流程应该如下

1. 扫描 `存储卡/tencent/MicroMsg/用户ID/image2/` 文件夹，得到所有原图的路径
2. 根据原图的文件名，计算缩略图的路径
3. 判断缩略图是否存在，如果存在则跳过
4. 使用 `PIL` 生成缩略图
5. 保存缩略图到指定路径下

有了思路，就开始动工写代码了。具体代码如下

```python
# encoding: utf-8
import os
import os.path
import string
from PIL import Image
from PIL import ImageFile
from shutil import copyfile
import struct
import hashlib

ImageFile.LOAD_TRUNCATED_IMAGES = True
ln = ["FFD8FF","89504E47","4D4D"]#JPG、PNG、TIFF
# 遍历文件 suffixList = ['.jpg','.png'.....]
def getFileList(path, suffixList=[]):
    ls = []
    list_dirs = os.walk(path)
    for root, dirs, files in list_dirs:
        for f in files:
            fp = (os.path.join(root, f))
            if not (f.startswith(".")):#过滤掉隐藏文件
                if (len(suffixList) == 0):
                    ls.append(fp)
                else:
                    suffix = os.path.splitext(fp)[-1]
                    if (suffix in suffixList):
                        ls.append(fp)
    return ls
#遍历1级目录
def getFolderList(path):
    ls = []
    list_dirs = os.listdir(path)
    for d in list_dirs:
        ls.append(d)
    return ls

# 字节码转16进制字符串
def bytes2hex(bytes):
    num = len(bytes)
    hexstr = u""
    for i in range(num):
        t = u"%x" % bytes[i]
        if len(t) % 2:
            hexstr += u"0"
        hexstr += t
    return hexstr.upper()

#缩放比为4.8，任意一边大于500就压缩
def printImgSize(path):
    try:
        src = Image.open(path)
        srcwidth, srcheight = src.size
        thumb = Image.open(getThumbnailPathFromOrigin(path))
        width,height = thumb.size
        print ("缩略图 width %d,height %d 原图 width %d,height %d"%(width,height,srcwidth,srcheight))
    except Exception as e:
        pass

#根据原图获取缩略图
def getThumbnailPathFromOrigin(path):
    fpName = getFileNameFromPath(path)
    if(fpName.startswith("th")):#如果是缩略图就直接返回了
        return path
    fpPath = getFilePath(path)
    return fpPath+"th_"+fpName.split(".")[0]


```

```
#根据缩略图获取原图
def getOriginPathFromThumbnailPath(path):
    fpName = getFileNameFromPath(path)
    if not (fpName.startswith("th")):#如果不是缩略图就直接返回了
        return path
    fpPath = getFilePath(path)
    fpName = fpName.replace("th_","")
    return fpPath+fpName+".png"

#获取文件名称
def getFileNameFromPath(path):
    pathName = path.split("/")
    return pathName[len(pathName)-1]


#获取文件路径
def getFilePath(fullPath):
    index = fullPath.rfind("/")
    return fullPath[0:index+1]

#校验图片是否有效
def isImageValidate(path,head = []):
    try:
        f_hcode = ""
        for hcode in head:
            binfile = open(path, 'rb') # 必需二制字读取
            numOfBytes = len(hcode) / 2 # 需要读多少字节
            binfile.seek(0) # 每次读取都要回到文件头，不然会一直往后读取
            hbytes = struct.unpack_from("B"*numOfBytes, binfile.read(numOfBytes)) # 一个 "B"表示一个字节
            f_hcode = bytes2hex(hbytes)
            binfile.close()
            if(f_hcode == hcode):
                return True
    except Exception as e:
        print (e)
    return False


COUNTER = 0
#缩放比为4.8，任意一边大于500就压缩
def compressImg(path):
    global COUNTER
    try:
        saveName = getFileNameFromPath(path)
        if not(saveName.startswith("th")):#不是缩略图才处理
            src = Image.open(path)
            width, height = src.size
            if(width >= 500 or height >= 500):#需要压缩
                tWidth = (int)(width / 4.8)
                tHeight = (int)(height / 4.8)
                print ("-"*20)
                print ("compress image %s,width %d height %d thumbnail Width %d thumbnail Height %d"%
                    (path,width,height,tWidth,tHeight))
                size = tWidth,tHeight
                src.load()
                src.thumbnail(size)
                saveName = getFileNameFromPath(path).split(".")[0]
                fileType = "JPEG"
                if(isImageValidate(path,["FFD8FF","4D4D"])):
                    fileType = "JPEG"
                elif(isImageValidate(path,["89504E47"])):
                    fileType = "PNG"
                savePath = getFilePath(path) + "th_"+saveName
                background = Image.new("RGBA", src.size, (255, 255, 255,255))
                if(len(src.split()) > 3):#处理透明通道
                    background.paste(src, mask=src.split()[3]) # 3 is the alpha channel
                else:
```

```
                    background.paste(src)
                print ("save to %s \n"%savePath)
                fileType = "PNG"
                background.save(savePath, fileType, quality=100)
            else:#不需要压缩，复制到新目录中
                savePath = getFilePath(path) + "th_" + saveName
                dst = savePath.split(".")[0]#去掉后缀
                copyfile(path, dst)
            COUNTER+=1
    except Exception as e:
        print (e)
        # raise e
        return False
    return True

#清空损坏的缩略图
def clearDeformThumbnial(ls):
    counter = 0
    for x in ls:
        thumbPath = getThumbnailPathFromOrigin(x)
        if not (isImageValidate(thumbPath,ln)):
            try:
                os.remove(thumbPath)
                counter+=1
                print ("deform image => %s,delete it"%thumbPath)
            except Exception as e:
                print (e)

    print ("Total remove %d"%counter)

#打印损坏的图片，并复制到deform文件夹中
def printDefromImage(ls):
    counter = 0
    for x in ls:
        if not (isImageValidate(x,ln)):
            counter += 1
            print ("deform image => %s"%x)
            dstPath = "deform/"+getFileNameFromPath(x)
            copyfile(x,dstPath)
    print ("Total deform %d"%counter)

#复制缩略图到thumbnail中
def copyThumbnial(ls):
    counter = 0
    try:
        for x in ls:
            name = getFileNameFromPath(x)
            if (name.startswith("th")):
                counter += 1
                print ("thumbnail image => %s"%name)
                dstPath = "thumbnail/"+name
                copyfile(x,dstPath)
    except Exception as e:
        print (e)
    print ("Total thumbnail %d"%counter)

#清空多余的图片
def clearReduceFile(ls):
    counter = 0
    for x in ls:
        fileName = getFileNameFromPath(x)
        if (fileName.startswith("th_th")):
            counter += 1
            print ("ReduceFile %s,remove"%x)
            os.remove(x)
    print ("Total %d remove %d"%(len(ls),counter))

```

```
#获取缩略图片的数量
def getThumbnialCount(ls):
    counter = 0
    for x in ls:
        fileName = getFileNameFromPath(x)
        if (fileName.startswith("th")):
            if(isImageValidate(x,ln)):
                counter += 1
    return counter

#获取原图的数量
def getOriginCount(ls):
    counter = 0
    for x in ls:
        fileName = getFileNameFromPath(x)
        if not (fileName.startswith("th")):
            if(isImageValidate(x,ln)):
                counter += 1
    return counter

#计算文件的MD5
def getMD5(fname):
    try:
        hash_md5 = hashlib.md5()
        with open(fname, "rb") as f:
            for chunk in iter(lambda: f.read(4096), b""):
                hash_md5.update(chunk)
        return hash_md5.hexdigest()
    except Exception as e:
        print e
    return ""


#计算重复文件的数量
def calcReduceFileCount():
    counter = 0
    ls = getFileList("image2")
    d = {}
    for x in ls:
        md5 = getMD5(x)
        # names = x.split("/")
        # md5 = names[len(names)-1]
        if(md5 in d):
            counter+=1
            n = d[md5]
            d[md5] = (n + "," + x)
        else:
            d[md5] = x
    for keys in d:
        if("," in d[keys]):
            print d[keys]
    print "Total reduce %d"%counter
#生成缩略图
def generateThumbnial():
    if not os.path.exists("deform"):
        os.makedirs("deform")
    if not os.path.exists("thumbnail"):
        os.makedirs("thumbnail")
    ls = getFileList("image2")
    fo = open("log.txt","w+")
    for x in ls:
        if(isImageValidate(x,ln)):#判断图片是否有效
            compressImg(x)
        else:#不是图片，写入日志
            dstPath = "deform/"+getFileNameFromPath(x)
            copyfile(x,dstPath)
            fo.write("No Image %s \n"%x)
```

```
268.
269.       ls = getFileList("image2")
270.       s = "total compress count: %d total list count: %d\n"%(COUNTER,len(ls))
271.       print (s)
272.       fo.write(s)
273.       s = "total %d,Thumbnail Count %d"%(getOriginCount(ls),getThumbnialCount(ls))
274.       print (s)
275.       fo.write(s)
276.       fo.close()
277.        #清空多余和损坏的文件
278.       clearReduceFile(getFileList("image2"))
279.       clearDeformThumbnial(getFileList("image2"))
280.       # #复制新的目录
281.       copyThumbnial(ls)
282.
283.   #push到手机里面
284.   #adb push image2/ /mnt/sdcard/tencent/MicroMsg/yourId/
285.   def pushToPhone():
286.       wechatID = "your id "
287.       ls = getFolderList("image2")
288.       fo = open("push.txt","w")
289.       count = 0
290.       for x in ls:
291.           cmd = "adb push image2/%s/ /mnt/sdcard/tencent/MicroMsg/%s/image2/"%(x,wechatID)
292.           res = "".join(os.popen(cmd).readlines())
293.           count+=1
294.           if("files pushed" in res):
295.               print "Push %s Success"%(x)
296.           else:
297.               err = "Push %s Failure\n"%(x)
298.               fo.write(err)
299.           print "pushing %d/%d"%(count,len(ls))
300.       fo.close()
301.
302.
303.   if __name__ == '__main__':
304.       # calcReduceFileCount()
305.       generateThumbnial()
306.       # pushToPhone()
```

重要提示：先备份你手机里的 `tencent/MicroMsg` 文件夹，避免意外情况导致你的文件丢失！

使用方法：

1. 手机连上数据线， `adb pull /mnt/sdcard/tencent/MicroMsg/yourId/` 到电脑上，假设存到 `micromsg` 下
2. 把脚本放在 `micromsg` 下，跟 `image2` 同级目录
3. 执行脚本，等待缩略图生成完毕
4. 用 `adb push image2/ /mnt/sdcard/tencent/MicroMsg/yourId/` 到手机上
    - 如果你的图片很多，直接执行 `adb push` 的话，如果中间某个文件push失败，adb会直接中断push。重新push又会重复传输已有图片。
    - 对于这种情况，可以在生成缩略图后，取消 `pushToPhone()` 的注释，并且在 `pushToPhone()` 里的 `wechatID` 填入你 `tencent/MicroMsg/` 下的微信ID。这样可以执行分文件push，通过 `push.txt` 记录已经push的文件，避免重复push。

微信　Python

What do you think?

1 Response

Upvote Funny Love Surprised

Angry Sad

0 Comments lsall.me Privacy Policy 1 Login

Recommend Tweet Share Sort by Best

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.