

KAKAO API

VISION

by python

PRESENT
7조

VISION

주요 특징

비전 API는 이미지 내부의 콘텐츠를 분석해서 얼굴, 상품, 성인물 여부, 문자를 판별하고 콘텐츠 중심의 썸네일이나 태그를 생성할 수 있습니다. 고급 인공지능 기술인 비전 API로 빠르고 정확하게 이미지를 분석하여 서비스에 맞게 다양하게 활용해보세요.

제공 기능

- 얼굴 검출: 이미지에서 얼굴을 탐색하고, 얼굴의 위치, 특징점, 각도, 성별, 나이 등을 분석합니다.
- 상품 검출: 이미지에 존재하는 상품의 위치와 종류를 검출합니다.
- 성인 이미지 판별: 이미지가 성인물인지 판별합니다.
- 썸네일 생성: 이미지의 콘텐츠를 잘 표현할 수 있는 위치를 찾아 요청한 크기의 썸네일(미리보기) 이미지로 생성합니다.
- 썸네일 검출: 이미지의 콘텐츠를 잘 표현할 수 있는 위치를 찾아 사용자의 눈길을 사로잡을 수 있는 썸네일을 제안합니다.
- 멀티 태그 생성: 이미지의 콘텐츠에 대한 태그를 생성합니다.
- OCR: 이미지에서 문자 영역을 검출하고 글자를 추출합니다.

앱 키

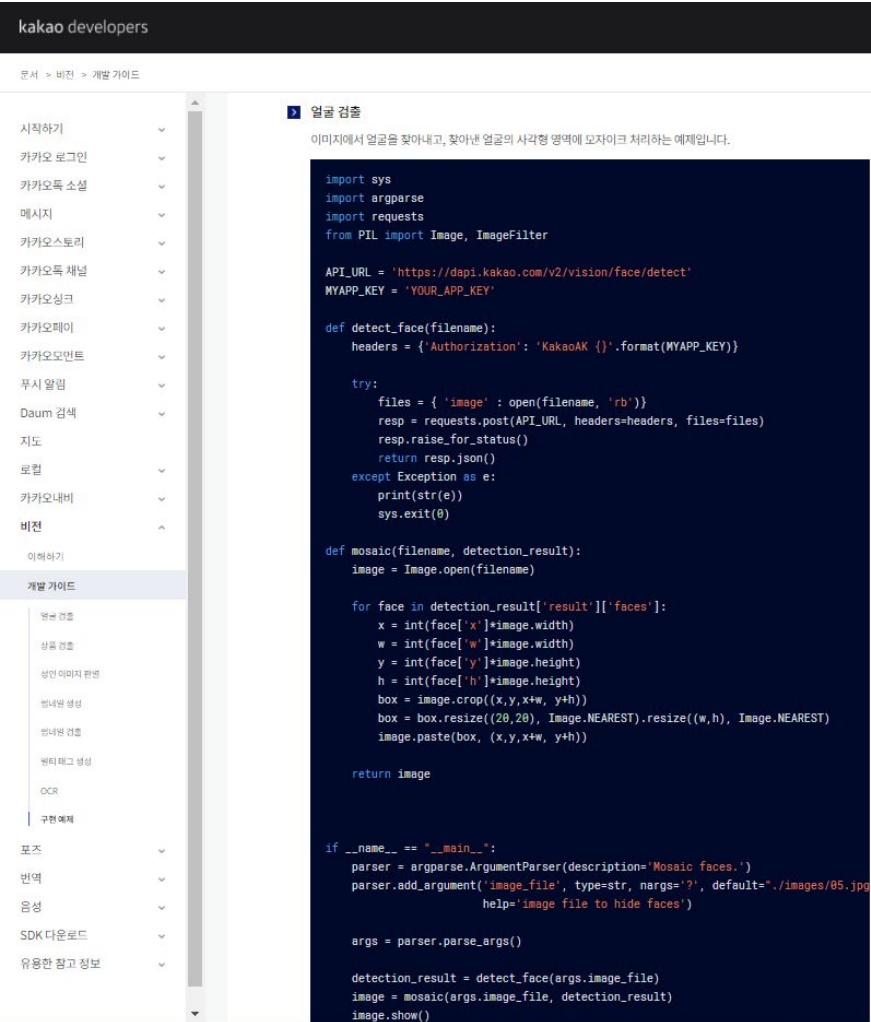
| | |
|--------------|----------------------------------|
| 네이티브 앱 키 | 02531eff729a87ecc6a83672992f8751 |
| REST API 키 | a2760a17994f03007a9c2615a230ec0c |
| JavaScript 키 | 13048da8d9b50ab83f67f5da0f4b4c96 |
| Admin 키 | 245710ebc391da96cdbb8b5db7ff6089 |

공통사항

- 서버에서 호출하는 경우 REST API 키를 사용해 **POST** 로 호출합니다.
 - 모든 API는 image 또는 image_url 중 하나의 값을 지정해야 합니다. 단, OCR API는 image만 지원합니다.
- image에 업로드되는 이미지와 image_url에 지정되는 이미지는 png 또는 jpg 포맷만 지원합니다. 단, OCR API는 bmp, dib, jpeg, jpg, jpe, jp2, png, webp, pbm, pgm, ppm, sr, ras, tiff, tif 포맷 또한 지원합니다.
- 이미지의 가로와 세로 길이는 2048px 이하여야 하고, 최대 용량은 2MB 입니다.
- image 파일을 업로드하는 경우는 Content-Type을 'multipart/form-data'로 요청해야 합니다.
- image_url로 호출하는 경우는 Content-Type을 'application/x-www-form-urlencoded'로 **POST** 되어야 합니다.

얼굴 검출

이미지에서 얼굴을 찾아내고, 찾아낸 얼굴의 사각형 영역에 모자이크 처리하는 예제 입니다.



```
work work02 work03
1 import sys
2 import argparse
3 import requests
4 from PIL import Image, ImageFilter
5 from statsmodels.stats.tests.test_influence import file_name
6
7 API_URL = 'https://dapi.kakao.com/v2/vision/face/detect'
8 MYAPP_KEY = 'KEY'
9
10 def detect_face(filename):
11     headers = {'Authorization': 'KakaoAK {}'.format(MYAPP_KEY)}
12
13     try:
14         files = {'image': open(filename, 'rb')}
15         resp = requests.post(API_URL, headers=headers, files=files)
16         resp.raise_for_status()
17         return resp.json()
18     except Exception as e:
19         print(str(e))
20         sys.exit(0)
21
22 def mosaic(filename, detection_result):
23     image = Image.open(filename)
24
25     for face in detection_result['result']['faces']:
26         x = int(face['x'] * image.width)
27         w = int(face['w'] * image.width)
28         y = int(face['y'] * image.height)
29         h = int(face['h'] * image.height)
30         box = image.crop((x, y, x+w, y+h))
31         box = box.resize((20, 20), Image.NEAREST).resize((w, h), Image.NEAREST)
32         image.paste(box, (x, y, x+w, y+h))
33
34     return image
35
36
37 if __name__ == "__main__":
38     parser = argparse.ArgumentParser(description='Mosaic faces.')
39     parser.add_argument('image_file', type=str, nargs='?', default='./images/05.jpg',
40                         help='image file to hide faces')
41
42     args = parser.parse_args()
43
44     detection_result = detect_face(args.image_file)
45     image = mosaic(args.image_file, detection_result)
46     image.show()
```



멀티태그 생성

이미지 내의 콘테츠에 대한 태그를 생성하는 API 입니다. 이미지의 카테고리를 분류할 수 있도록 도움을 줍니다.

kakao developers

문서 > 비전 > 개발 가이드

시작하기

카카오 로그인

카카오오픈소스

메시지

카카오스토리

카카오오픈채널

카카오싱크

카카오메이

카카오모먼트

푸시 알림

Daum 검색

지도

로컬

카카오내비

비전

이해하기

개발 가이드

원본 건물

상물 건물

상인 이미지 판별

판넬 생성

판넬 건물

멀티태그 생성

OCR

구현 예제

포즈

번역

Q&A

멀티태그 생성

이미지를 대표하는 태그를 추출해서 출력하는 예제입니다.

```
import sys
import argparse
import requests
from PIL import Image, ImageDraw, ImageFont
from io import BytesIO

API_URL = 'https://dapi.kakao.com/v2/vision/multitag/generate'
MYAPP_KEY = 'YOUR_APP_KEY'

def generate_tag(image_url):
    headers = {'Authorization': 'KakaoAK {}'.format(MYAPP_KEY)}

    try:
        data = { 'image_url' : image_url }
        resp = requests.post(API_URL, headers=headers, data=data)
        resp.raise_for_status()
        result = resp.json()[ 'result' ]
        if len(result[ 'label_kr' ]) > 0:
            if type(result[ 'label_kr' ][0]) != str:
                result[ 'label_kr' ] = map(lambda x: str(x.encode("utf-8")), result[ 'label_kr' ])
            print("이미지를 대표하는 태그는 '{}'입니다.".format(','.join(result[ 'label_kr' ])))
        else:
            print("이미지로부터 태그를 생성하지 못했습니다.")

    except Exception as e:
        print(str(e))
        sys.exit(0)

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='Classify Tags')
    parser.add_argument('image_url', type=str, nargs='?',
                        default="http://t1.daumcdn.net/alvo/vision/openapi/r2/images/08.jpg",
                        help='image url to classify')

    args = parser.parse_args()

    generate_tag(args.image_url)
```

work02 work03 C:\workspace_python\HELLOPYTHON

```
sys
import argparse
import requests
from PIL import Image, ImageDraw, ImageFont
from io import BytesIO

API_URL = 'https://dapi.kakao.com/v2/vision/multitag/generate'
MYAPP_KEY = 'KEY'

def generate_tag(image_url):
    headers = {'Authorization': 'KakaoAK {}'.format(MYAPP_KEY)}


    try:
        data = { 'image_url' : image_url }
        resp = requests.post(API_URL, headers=headers, data=data)
        resp.raise_for_status()
        result = resp.json()[ 'result' ]
        if len(result[ 'label_kr' ]) > 0:
            if type(result[ 'label_kr' ][0]) != str:
                result[ 'label_kr' ] = map(lambda x: str(x.encode("utf-8")), result[ 'label_kr' ])
            print("이미지를 대표하는 태그는 '{}'입니다.".format(','.join(result[ 'label_kr' ])))
        else:
            print("이미지로부터 태그를 생성하지 못했습니다.")

    except Exception as e:
        print(str(e))
        sys.exit(0)

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='Classify Tags')
    parser.add_argument('image_url', type=str, nargs='?',
                        default="https://t1.daumcdn.net/cj/08.jpg",
                        help='image url to classify')

    args = parser.parse_args()

    generate_tag(args.image_url)
```



Markers Properties Servers Data Source Explorer Console Progress

<terminated> work02.py [D:\Anaconda3\python.exe]

이미지를 대표하는 태그는 "가구, 의자, 테이블"입니다.

상품 검출

이미지 내에 존재하는 상품의 위치와 종류를 반환합니다.



상품 검출

이미지로부터 상품 영역을 사각형으로 검출하고 검출된 상품 이름을 보여주는 예제입니다.

```
import sys
import argparse
import requests
from PIL import Image, ImageDraw, ImageFont
from io import BytesIO

API_URL = 'https://dapi.kakao.com/v2/vision/product/detect'
MYAPP_KEY = 'YOUR_APP_KEY'

def detect_product(image_url):
    headers = {'Authorization': 'KakaoAK {}'.format(MYAPP_KEY)}

    try:
        data = { 'image_url' : image_url }
        resp = requests.post(API_URL, headers=headers, data=data)
        resp.raise_for_status()
        return resp.json()
    except Exception as e:
        print(str(e))
        sys.exit(0)

def show_products(image_url, detection_result):
    try:
        image_resp = requests.get(image_url)
        image_resp.raise_for_status()
        file_jpgdata = BytesIO(image_resp.content)
        image = Image.open(file_jpgdata)
    except Exception as e:
        print(str(e))
        sys.exit(0)

    draw = ImageDraw.Draw(image)
    for obj in detection_result['result']['objects']:
        x1 = int(obj['x1']*image.width)
        y1 = int(obj['y1']*image.height)
        x2 = int(obj['x2']*image.width)
        y2 = int(obj['y2']*image.height)
        draw.rectangle([(x1,y1), (x2, y2)], fill=None, outline=(255,0,255))
        draw.text((x1+5,y1+5), obj['class'], (255,0,0))
    del draw

    return image

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='Detect Products.')
    parser.add_argument('image_url', type=str, help='image url to show product')
    args = parser.parse_args()

    detection_result = detect_product(args.image_url)
    image = show_products(args.image_url, detection_result)
    image.show()
```

```
work work02 *work03 C:\workspace_python\WHELLOPYTHON
7 API_URL = 'https://dapi.kakao.com/v2/vision/product/detect'
8 MYAPP_KEY = 'KEY'
9
10 def detect_product(image_url):
11     headers = {'Authorization': 'KakaoAK {}'.format(MYAPP_KEY)}
12
13     try:
14         data = { 'image_url' : image_url }
15         resp = requests.post(API_URL, headers=headers, data=data)
16         resp.raise_for_status()
17         return resp.json()
18     except Exception as e:
19         print(str(e))
20         sys.exit(0)
21
22 def show_products(image_url, detection_result):
23     try:
24         image_resp = requests.get(image_url)
25         image_resp.raise_for_status()
26         file_jpgdata = BytesIO(image_resp.content)
27         image = Image.open(file_jpgdata)
28     except Exception as e:
29         print(str(e))
30         sys.exit(0)
31
32     draw = ImageDraw.Draw(image)
33     for obj in detection_result['result']['objects']:
34         x1 = int(obj['x1']*image.width)
35         y1 = int(obj['y1']*image.height)
36         x2 = int(obj['x2']*image.width)
37         y2 = int(obj['y2']*image.height)
38         draw.rectangle([(x1,y1), (x2, y2)], fill=None, outline=(255,0,255))
39         draw.text((x1+5,y1+5), obj['class'], (255,0,0))
40     del draw
41
42     return image
43
44 if __name__ == "__main__":
45     parser = argparse.ArgumentParser(description='Detect Products.')
46     parser.add_argument('image_url', type=str, help='image url to show product')
47     args = parser.parse_args()
48
49     detection_result = detect_product(args.image_url)
50     image = show_products(args.image_url, detection_result)
51     image.show()
```