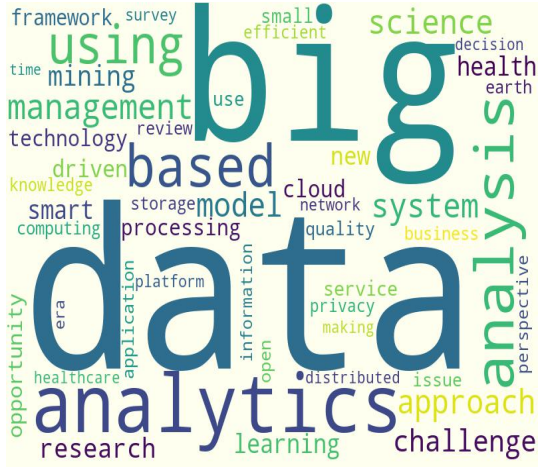
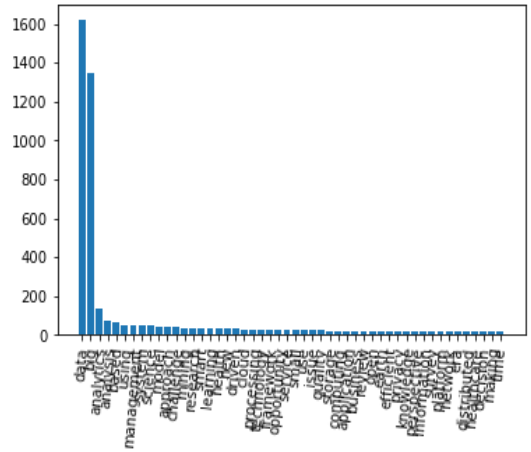
NLP 기반 Word Cloud

Overview

많은 양의 데이터가 발생하는 요즘 데이터를 가공하고 처리하는 것 뿐만 아니라 그것을 보기 쉽게 보여주기 위한 방식이 중요해지고 있다. 이 프로그램은 Word Cloud를 통하여 해당 단어가 해외 논문에서 얼마나 많이 사용 되었는지에 대한 빈도수를 시각화 하는 프로그램이다.



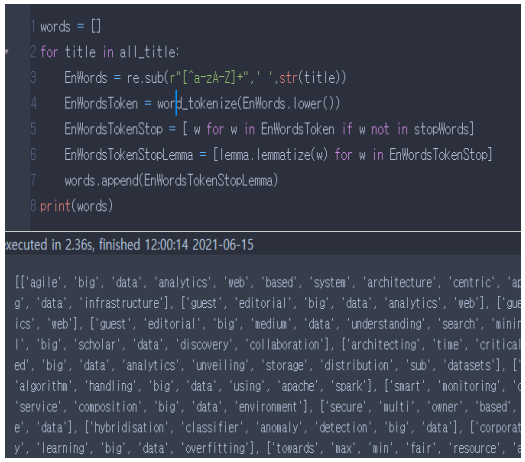


Technology

Language : Python 3.8 (anaconda)
IDE : jupyter Notebook
Modules : NLTK, MatPlotlib, Pandas

Result

Stop words를 제외한 나머지로 토큰화
Word Cloud, Matplot을 통하여 이미지를 생성



```
words = []
for title in all_title:
    EnWords = re.sub(r"[^a-zA-Z]+"," ",str(title))
    EnWordsToken = word_tokenize(EnWords.lower())
    EnWordsTokenStop = [ w for w in EnWordsToken if w not in stopWords]
    EnWordsTokenStopLemma = [lemma.lemmatize(w) for w in EnWordsTokenStop]
    words.append(EnWordsTokenStopLemma)
print(words)
```

```
executed in 2.36s, finished 12:00:14 2021-06-15
```

```
[['agile', 'big', 'data', 'analytics', 'web', 'based', 'system', 'architecture', 'centric', 'ap
g', 'data', 'infrastructure'], ['guest', 'editorial', 'big', 'data', 'analytics', 'web'], ['gue
ics', 'web'], ['guest', 'editorial', 'big', 'medium', 'data', 'understanding', 'search', 'minin
l', 'big', 'scholar', 'data', 'discovery', 'collaboration'], ['architecting', 'time', 'critical
ed', 'big', 'data', 'analytics', 'unveiling', 'storage', 'distribution', 'sub', 'datasets'], ['
'algorithm', 'handling', 'big', 'data', 'using', 'apache', 'spark'], ['smart', 'monitoring', 'c
'service', 'composition', 'big', 'data', 'environment'], ['secure', 'multi', 'owner', 'based',
e', 'data'], ['hybridisation', 'classifier', 'anomaly', 'detection', 'big', 'data'], ['corporat
y', 'learning', 'big', 'data', 'overfitting'], ['towards', 'max', 'min', 'fair', 'resource', 'a
```

단어 전처리 - 토큰화

```
데이터 전처리
• stopwrod 불러오기
• 만약 에러 발생시 :
```

```
import nltk
nltk.download('stopwords')
```

```
stopWords = set(stopwords.words('english'))
lemma = WordNetLemmatizer()
```

```
executed in 42ms, finished 12:00:12 2021-06-15
```

```
데이터 전처리
• 단어 단위로 자르기
• 에러발생시
```

```
import nltk
nltk.download('punkt')
nltk.download('wordnet')
```

```
words = []
for title in all_title:
    EnWords = re.sub(r"[^a-zA-Z]+"," ",str(title))      # 알파벳을 제외한 나머지를 공백으로 치환
    EnWordsToken = word_tokenize(EnWords.lower())       # 소문자로 통일 후 단어별로 자름
    EnWordsTokenStop = [ w for w in EnWordsToken if w not in stopWords]  # 자른단어가 종료문자에 있지않으면 리스트에 추가
    EnWordsTokenStopLemma = [lemma.lemmatize(w) for w in EnWordsTokenStop]  # 표제어 추출
    words.append(EnWordsTokenStopLemma)
print(words)
```

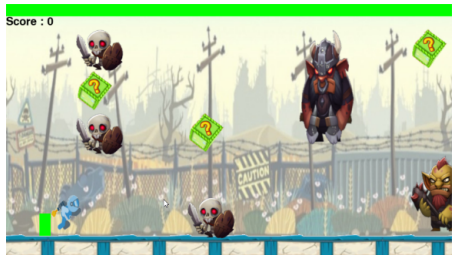
단어 전처리 - 토큰화

Python 기반 슈팅게임(가제)

Overview

아무리 기능이 좋은 프로그램이라도 User Interface/User eXperience (UI/UX)가 안좋으면 사양되는 것이 현실이다. 이 프로그램을 통하여 GUI에서 UI/UX를 구현하는 것이 목표이다.

첫 실행 화면



인게임 화면

Technology

Language : Python 3.8 (anaconda)
IDE : jupyter Notebook
Modules : PyGame

캐릭터 클래스

화면 출력 함수

Result

시간에 따른 이동속도 변화
이미지 및 사운드(배경, 피격등) 적용

실시간 객체 탐지

Overview

AI가 발전함에 따라서 학습 및 테스트 데이터의 전처리 및 실생활에 적용에 사용되는 영상처리의 기술의 중요성도 증가가 되고있다. 이 프로그램은 OpenCV와 학습된 데이터를 활용하는 영상처리의 예를 보여준다.

Technology

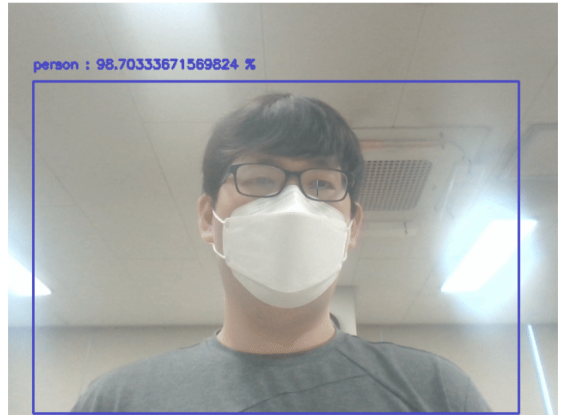
Language : Python 3.8 (anaconda)
IDE : jupyter Notebook
Modules : OpenCV, numpy

Result

Webcam/카메라의 촬영되는 객체 구분
구분한 객체가 프레임위치 표시

```python
while done:
    ret, image = cap.read()
    if ret :
        (h,w) = image.shape[:2]
        blob = cv2.dnn.blobFromImage(cv2.resize(image,(300,300)),
                                     0.007843,
                                     (300,300),
                                     127.5)
        net.setInput(blob)
        detections = net.forward()

        for i in np.arange(0,detections.shape[2]):
            confidence = detections[0,0,i,2]
            if confidence > 0.2 :
                idx = int(detections[0,0,i,1])
                box = detections[0,0,i,3:7] *np.array([w,h,w,h])
                (sx,sy,ex,ey) = box.astype('int')

                label = '{} : {} %'.format(CLASSES[idx],confidence *100)
                cv2.rectangle(image,(sx,sy),(ex,ey), COLORS[idx],2)
                y = sy -15 if sy -15 > 15 else sy +15
                cv2.putText(image,
                            label,
                            (sx,y),
                            cv2.FONT_HERSHEY_SIMPLEX,
                            0.5,
                            COLORS[idx],
                            2)
        cv2.imshow('Detection',image)
        key = cv2.waitKey(delay)
        if key == ord('q'):
            break
cap.release
cv2.destroyAllWindows()
```

Web cam 데이터를 받아와서 처리



결과 화면

```python
CLASSES = ['background', 'aeroplane','bicycle', 'bird','boat','bottle','bus','car',
           'chair','cow','diningtable', 'dog', 'horse','motobike',
           'pottedplant', 'person', 'sheep',  'sofa', 'trian', 'tvmonitor']

COLORS = np.random.uniform(0,255,size = (len(CLASSES),3))
```

```
ecuted in 14ms, finished 09:30:41 2021-06-23
```

```python
net = cv2.dnn.readNetFromCaffe('./deploy.prototxt',
                               'mobilenet_iter_73000.caffemodel')
```

분류할 클래스들 설정과
가중치를 로드

Webcam을 활용한 얼굴인식

Overview

한 종류의 데이터 셋과 한가지의 모델에서 100% 모든 것을 구분할 수 없고, 구분한다 하더라도 정확도는 낮다. 이러한 문제를 해결하는 방안 중 앙상블이 존재한다. 이 프로그램은 학습된 데이터들의 앙상블을 활용하여 사람의 얼굴을 인식하는 프로그램을 나타낸다.

Technology

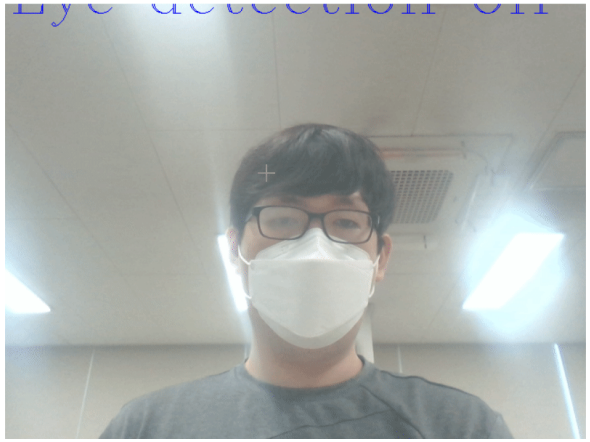
Language : Python 3.8 (anaconda)
IDE : jupyter Notebook
Modules : OpenCV, Numpy, Matplotlib

Result

Webcam/카메라의 촬영되는 객체 구분
구분한 객체가 프레임위치 표시

```
    cv2.putText(frame, info,(5,10),font,2,(255,0,0),1)
    for ( x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y),(x+w,y+h), (255,0,0),2)
        cv2.putText(frame,'Deteted Face',(x-5,y-5), font,1,fontColor,fontScale)
        if eye_detect :
            roi_gray = gray[y:y+h,x:x+w]
            roi_color = frame[y:y+h, x:x+w]
            eyes = eye_cascade.detectMultiScale(roi_gray,1.1, 3)
            for (ex,ey,ew,eh) in eyes:
                cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
    cv2.imshow('frame',frame)
```

화면에 출력



결과 화면

```
face_cascade = cv2.CascadeClassifier('./haarcascade_frontface.xml')
eye_cascade = cv2.CascadeClassifier('./haarcascade_eye.xml')

try :
    cap = cv2.VideoCapture(0)
except :
    print("Camera Loading Failed")
    return
```

가중치 및 Webcam 로드

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray,1.3,5)
```

Webcam의 프레임을 전처리 및 판별

YOLO을 활용한 물체 인식

Overview

YOLO는 기존의 복잡한 객체 검출 프로세스를 하나의 회귀 문제로 바꾸어 기존의 객체 인식보다 빠르고 보다 실시간으로 처리할 수 있다.
이 프로그램은 YOLO를 활용하여 객체를 구분하는 것을 목표로 한다.

Technology

Language : Python 3.8 (anaconda)
IDE : jupyter Notebook
Modules : OpenCV, Numpy, Matplotlib

Result

이미지/웹캠 상에 존재하는 객체를 구분
구분한 객체가 이미지/웹캠에 위치 표시
웹캠으로 촬영한 영상 avi확장자로 저장

```
net = cv2.dnn.readNet('./YOLO/yolov3.weights','./YOLO/yolov3.cfg')
classes = []
with open('./YOLO/coco.names','r') as f:
    classes = [line.strip() for line in f.readlines()]
layer_name = net.getLayerNames()
output_layers = [layer_name[i[0] -1 ] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0,255,size = (len(classes),3))
```

가중치 및 Webcam 로드

이미지 판별

```
for out in outs:
    for detection in out :
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.5 :
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)

            w = int(detection[2] * width)
            h = int(detection[3] * height)

            x = int( center_x - w/2)
            y = int( center_y - h/2)

            boxes.append([x,y,w,h])
            confidences.append(float(confidence))
            class_ids.append(class_id)
```

추출한 외곽선 표시

결과 화면

명함 인식

Overview

YOLO는 기존의 복잡한 객체 검출 프로세스를 하나의 회귀 문제로 바꾸어 기존의 객체 인식보다 빠르고 보다 실시간으로 처리할 수 있다.
이 프로그램은 YOLO를 활용하여 객체를 구분하는 것을 목표로 한다.

Technology

Language : Python 3.8 (anaconda)
IDE : jupyter Notebook
Modules : OpenCV, Numpy, Matplotlib

Result

이미지에 존재하는 객체를 구분
구분한 객체가 이미지에 어느 부분에 존재하는지 표시

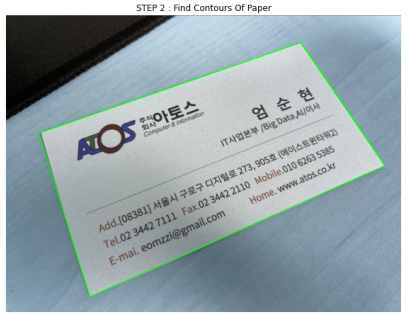
```
r = 800.0/image.shape[0]
dim = (int(image.shape[1] * r) ,800)

image = cv2.resize(image,dim,interpolation = cv2.INTER_AREA)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray,(3,3),0)
edged = cv2.Canny(gray,75,200)
```

외곽선 검출

외곽선 검출 출력 화면

```
screenCnt = np.array([])

ret,thresh = cv2.threshold(edged.copy(),127,255,0)

temp = cv2.findContours(thresh, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
cnts = temp[0] if len(temp) == 2 else temp[1]

cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:5]

for c in cnts:
    peri = cv2.arcLength(c, True)
    approx = cv2.approxPolyDP(c , 0.02 * peri, True)
    if len(approx) == 4:
        screenCnt = approx
        break
```

외곽선 표시



외곽선 표시 화면

```
def order_points(pts):
    rect = np.zeros((4,2),dtype ="float32")
    s = pts.sum(axis = 1)

    rect[0] = pts[np.argmin(s)]
    rect[2] = pts[np.argmax(s)]

    diff = np.diff(pts,axis = 1)
    rect[1] = pts[np.argmin(diff)]
    rect[3] = pts[np.argmax(diff)]

    return rect
rect = order_points(screenCnt.reshape(4,2)/r)
(tl,tr,br,bl) = rect

w1 = abs(br[0]-bl[0])
w2 = abs(tr[0]-tl[0])
h1 = abs(tr[1]-br[1])
h2 = abs(tl[1]-bl[1])

maxwidth = max([w1,w2])
maxheight = max([h1,h2])

dst = np.float32([[0,0],[maxwidth-1,0],[maxwidth-1,maxheight-1],[0,maxheight-1]])

M = cv2.getPerspectiveTransform(rect, dst)
img = orig.copy()


warped = cv2.warpPerspective(img, M, (int(maxwidth), int(maxheight)))
```

외곽선에 맞춰서 기하변환

결과 화면