

특정 이미지와 관련된 인스타그램 해시태그를 추천

에이콘 아카데미 5팀

팀장 : 조성준

팀원 : 김다해, 서희진, 이준형, 최인권, 이시우

프로젝트 진행순서

01

데이터셋
만들기
Roboflow란?
Weight파일 도출

02

Yolov5
이미지 객체인식
Yolov5를 선택한 이유
Yolov5로 이미지 객체 인식, 분석

03

인스타그램
해시태그 크롤링
Selenium이란?
데이터 크롤링하여 파일로 저장

04

Word2Vec
키워드 도출
Word2Vec란?
유사도 높은 단어 시각화

05

LDA
키워드 도출
LDA란?
유사도 높은 단어 시각화

06

장고
웹 출력
장고를 통해서 제작한
웹 페이지 설명



이시우

크롤링 코드 제작,
웹구현,
디버깅



최인권

단어 크롤링 및
워드투벡터
모델생성,
시각화



이준형

이미지 웹 크롤링
데이터셋 라벨작업,
수정, 제작
colab에서 가중치
파일 생성
모델학습 및 분석



서희진

LDA모델을 이용한
유사단어 추출
알고리즘 구현



김다해

Yolov5 모델 데이터셋
콜랩에서 제작
코드 생성,
이미지 객체 인식 코드
생성,
장고부트스트랩
레이아웃 제작,
ppt제작, 장고 메인
코드 구현



조성준

yolov5 데이터셋 제작,
웹크롤링, yolov5
모델링 및 분석,
colab을 이용해
best.pt파일 도출,
yolov5 분석자료
웹 출력



django



YOLO v5



01

데이터셋
만들기

Roboflow란?

Roboflow는 데이터셋을 구성하는 서비스를 제공합니다. 다양한 기능을 제공하지만 그 중 특히 유용한 기능 중 하나는 원하는 데이터셋 형식으로 export할 수 있다는 점입니다.

999pro

Last Upload
3 hours ago
981 images added

Dataset Size
981 Images
Augmentation Disabled

Annotations
5t
Object Detection

Images



[View unannotated images \(2\)](#) [View all images \(981\)](#)

Train/Test Split

Your images are split at upload time. [Learn more.](#)

Edit Splits



+ Add Augmentation Step

Preprocessing Options

Applied to all images in dataset

+ Add Preprocessing Step

Auto-Orient
Remove

Computer Vision Datasets

Roboflow hosts free public computer vision datasets in many popular formats (including CreateML JSON, COCO JSON, Pascal VOC XML, YOLO v3, and Tensorflow TFRecords). For your convenience, we also have downsized and augmented versions available.

If you'd like us to host your dataset, please [get in touch](#).

Aquarium Dataset
 Object Detection (Bounding Box)
638 Images | 3 exports | Last updated 4 days ago

Roboflow에서는 데이터셋을 무료로 제공합니다.
<https://roboflow.com/>

2. 라벨 지정

라벨로 설정할 부분을 네모로 지정하여 좌표값을 만들어준다.



3. Train/Valid/Test Split 지정

Train Test비율을 지정해줍니다.



1. 이미지 수집

데이터를 수집합니다.



4. YOLO v5 Torch Export

데이터셋을 학습시키기 위해 export시켜준후, zip url을 만들어줍니다.

Export

Format: YOLO v5 PyTorch

curl -L "https://app.roboflow.com/.../unzip roboflow.zip" rm roboflow.zip

Your Download Code

Download zip to computer

Paste this snippet into a notebook to download and unzip your dataset:

curl -L "https://app.roboflow.com/.../unzip roboflow.zip" rm roboflow.zip

Warning: Do not share this link beyond your team, it contains a private key that is tied to your Roboflow account. Acceptable use policy applies.

- Roboflow에서 만든 데이터셋(zip url)을 콜랩에서 풀어 학습을 시켜줍니다.

CO yolov5 0413.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말 변경사항이 저장되지 않음

人脉 公유

+ 코드 + 텍스트 Drive로 복사

연결 ▾ 수정

curl -L "https://app.roboflow.com/ds/cuCYxf/PNy?key=S1vuiNsQEq" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip

extracting: test/images/cake--3_.jpg.rf.a640db2124897dc2da88d141dd6b64ec.jpg
extracting: test/images/cake--33_.jpg.rf.b09c192ce7966e0fab4fc9ccc366bdb9.jpg
extracting: test/images/cake--3_.jpg.rf.2602edd7fcb925cc29f41d8334f57d79.jpg
extracting: test/images/cake--3_.jpg.rf.31c5473731155bc682d6fd4399cec225.jpg
extracting: test/images/cake--3_.jpg.rf.169dfc400640dda1e1a5ea2f856cc2fb.jpg
extracting: test/images/cake--3_.jpg.rf.9c4b25ecc5428de8b0ac42b5dc2bc688.jpg
extracting: test/images/cake--3_.jpg.rf.a51d08278fb4bc06584e4b59ef1ff95d.jpg
extracting: test/images/cake--3_.jpg.rf.075542bddba084c1be7d98ccbba151c.jpg
extracting: test/images/cake--3_.jpg.rf.d4701122773b8cdc79f9f2bee23d3f8f.jpg
extracting: test/images/cake--3_.jpg.rf.85c98ecd77b2067f009597e6ed09f597.jpg
extracting: test/images/cake--3_.jpg.rf.6f6d82ah2e6a2470740c38261616094.jpg

Your Download Code

Jupyter Terminal Raw URL

Paste this snippet into a notebook to download and unzip [your dataset](#):

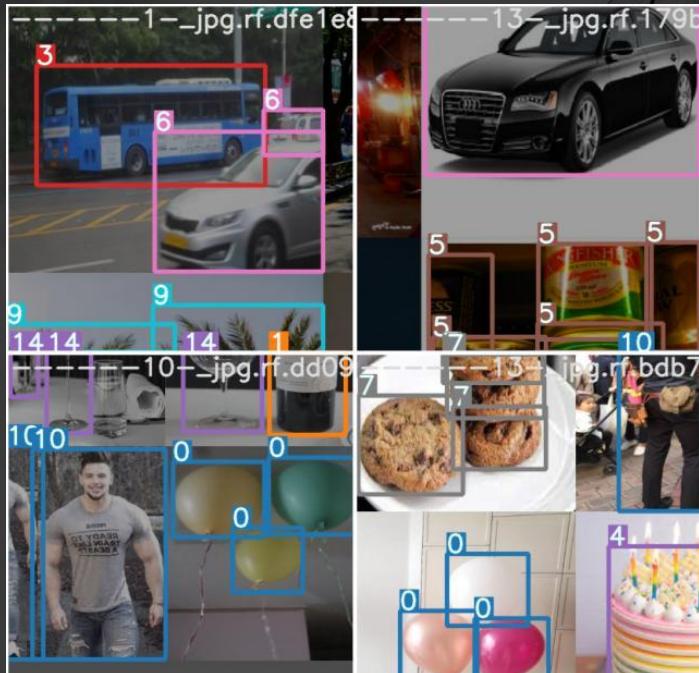
```
curl -L "https://app.roboflow.com/ds/cuCYxf/PNy?key=S1vuiNsQEq" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
```

Warning: Do not share this link beyond your team, it contains a private key that is tied to your Roboflow account. Acceptable use policy applies.

Done

<https://colab.research.google.com/drive/1IWYDZNOAQyPjv7wZOs0YYgGbmarEOfqG?usp=sharing>

Train

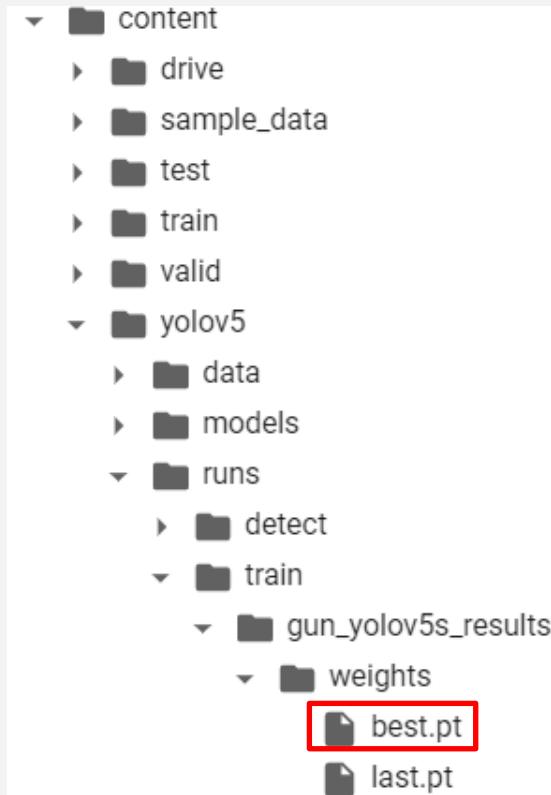


Feature : 이미지, **Label** : 좌표값

Test



Train을 통해 이미지를 인식



Best.pt란?

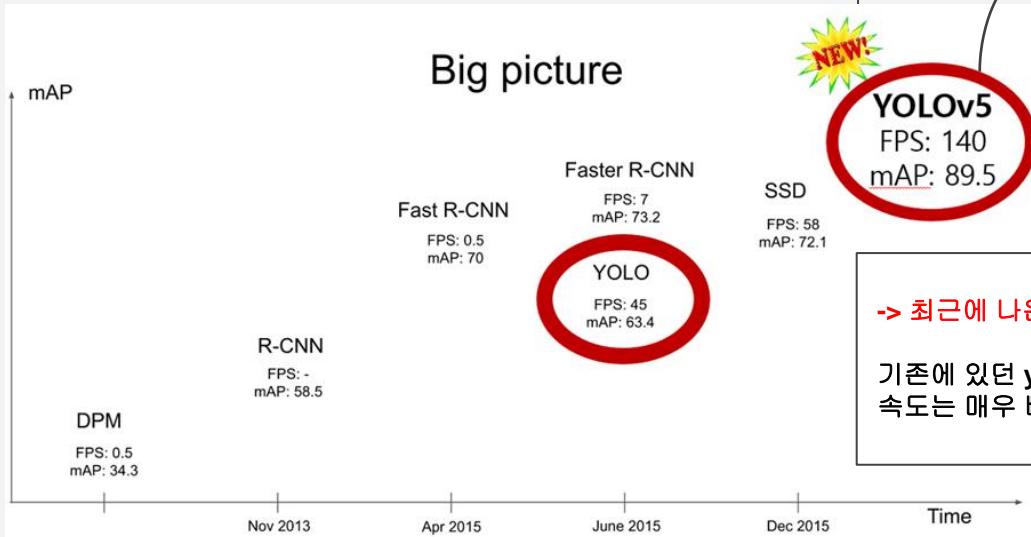
Best.pt파일은 weight file 파일입니다.
학습된 내용이 담겨져 객체인식할때 필요합니다.
weight file은 best.pt와 last.pt 가 나오는데 best.pt는
가장 성능이 좋은 weight 파일이고, last.pt는 최신 weight
파일입니다.

Yolov5 이미지 객체 인식

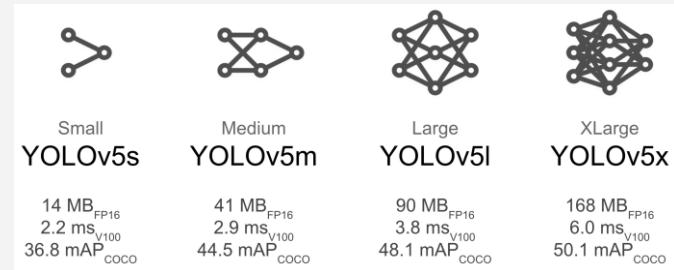
02

Yolov5를 선택한 이유는?

YOLO는 You Only Look Once의 약자로, 객체탐지분야에서 많이 알려진 모델이다. 현재 YOLO, YOLOv2, YOLOv3, YOLOv4, YOLOv5까지 나왔습니다. 저희팀에서 사용한 YOLOv5는 2020년 6월에 출시되었습니다.



FPS : 초당 프레임 수
mAP : 평균 정밀도



YOLOv5는 다른 yolo 모델들과 달리 4가지의 버전이 있다.
YOLOv5s = 가장 가벼운 모델
YOLOv5x = 가장 무거운 모델
당연하게 s의 성능이 제일 낮지만 FPS는 가장 높고,
X의 성능이 제일 높지만 FPS는 가장 낮다.

-> 최근에 나온 YOLOv5는 속도와 정확도에서 압도적인 성능을 발휘하고 있다.

기존에 있던 yolov3는 FPS는 높은 반면 mAP는 낮은 모델이었다.
속도는 매우 빠른 반면 정확도는 Faster R-CNN 보다 약간 낮다.

웹 상에서 업로드한 이미지



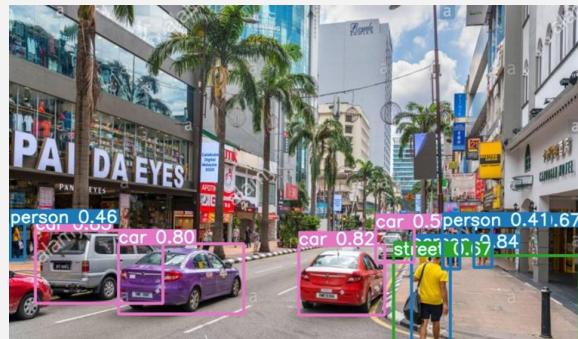
Yolov5로 객체 인식된 이미지



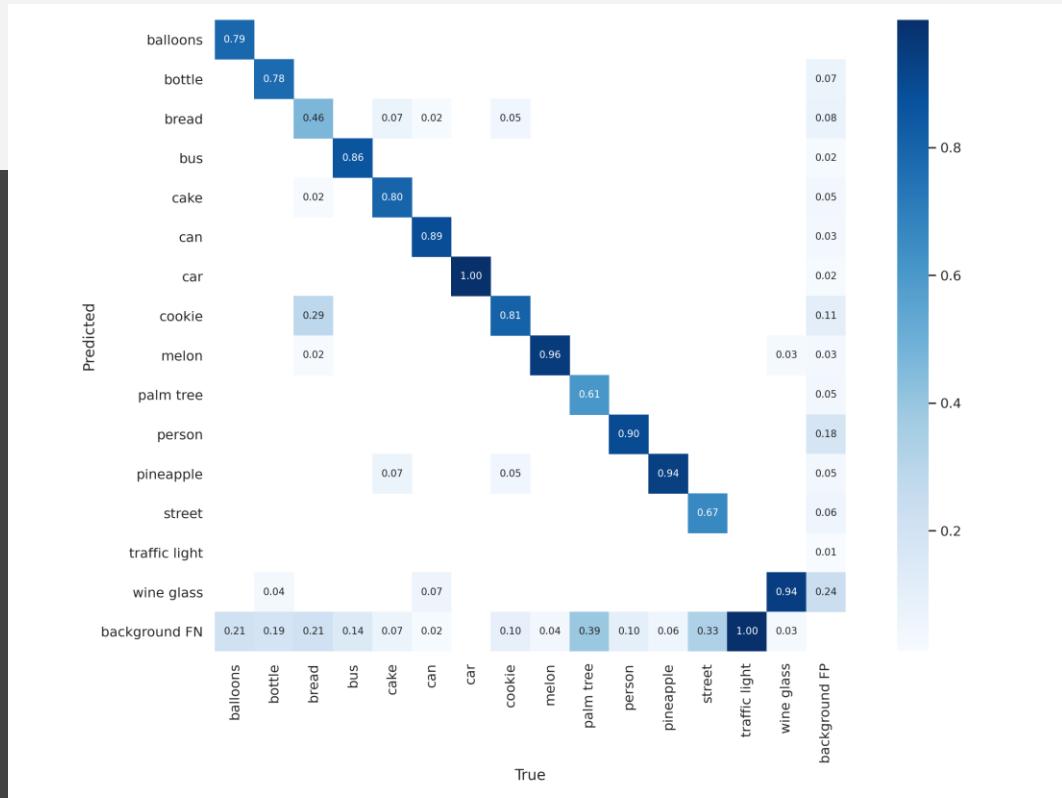
Yolov5로 객체 인식을 하게 되면 오른쪽 화면처럼 보이게 됩니다.

인식된 이름이 ['balloons', 'bottle', 'bread', 'cake', 'can', 'wine glass']와 같이 리스트로 받아지며 if문을 통해 ['풍선', '병', '빵', '케잌', '캔', '와인잔'] 한글 리스트 값으로 변환해주어 사용합니다.

객체 인식된 또 다른 이미지



학습시킨 Yolov5 confusion_matrix



Training을 통한 prediction 성능을 측정하기 위해 예측 value와 실제 value를 비교하기 위한 표이다.
데이터셋에 대한 분류 모델의 성능을 설명해줌.

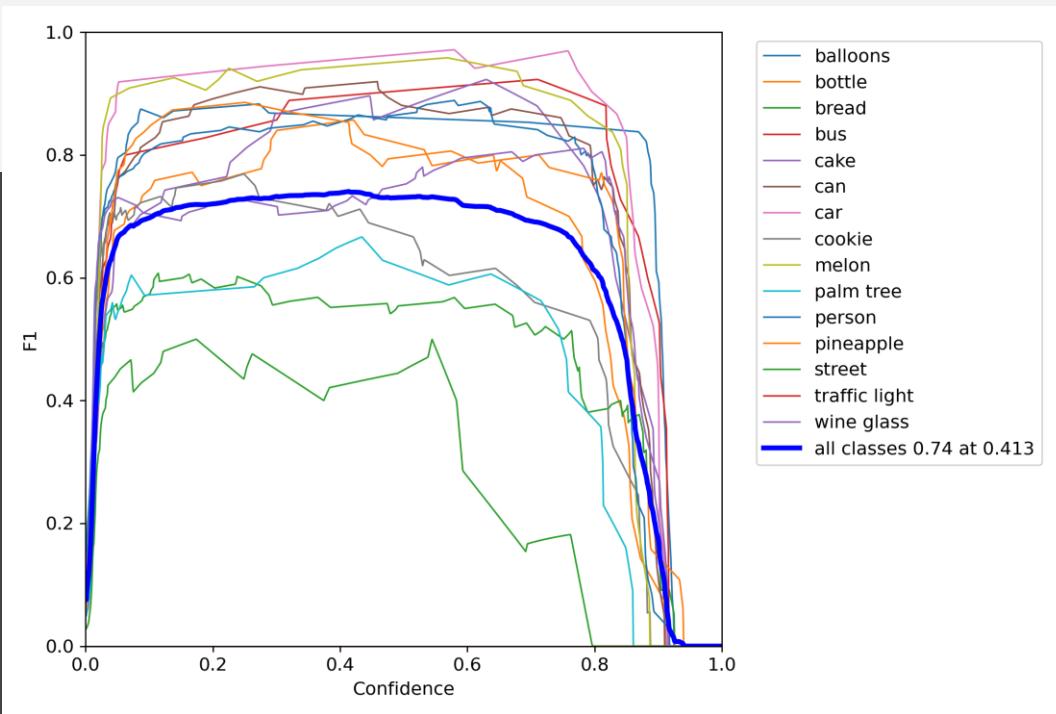
FP : 틀린 검출

FN : 검출되어야 할 것이 검출되지 않음.

precision : 정밀도 = 옳게 검출한 비율

Recall : 검출율(재현율) = 실제 옳게 검출된 결과물 중에서 옳다고 예측한 것의 비율

학습시킨 Yolov5 F1_curve



F1-Curve : F1-Score와 confidence를 가지고 나타낸
그래프

F1-Score : Precision과 Recall의 조화 평균

사용 이유 : Accuracy가 아닌 F1 Score를 쓰는 경우는
데이터가 불균형(Imbalance) 할 때이다.

Confidence(신뢰구간) = 검출한 것에 대해 알고리즘이¹
얼마나 정확하다고 생각하는지 알려주는 값

Confidence Threshold (기준 신뢰도, objectness)

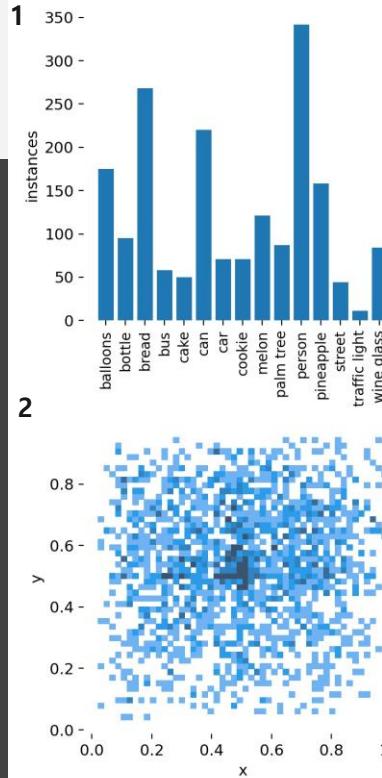
Confidence Threshold 가 변하면 Recall과 Precision 이
크게 변한다.

Confidence Threshold = 0.1 인 경우 영상 전체에서
후보 박스가 가득 나타나게 된다.

Confidence Threshold = 0.99 인 경우 정말 오브젝트가
있는데 확실한 후보만 남는다.

Precision-recall 곡선과 , Average precision은
컴퓨터비전(Computer Vision)에서 물체 검출(Object
Detection) 알고리즘 성능 평가를 위해 사용되는
방법이다.

학습시킨 Yolov5 labels

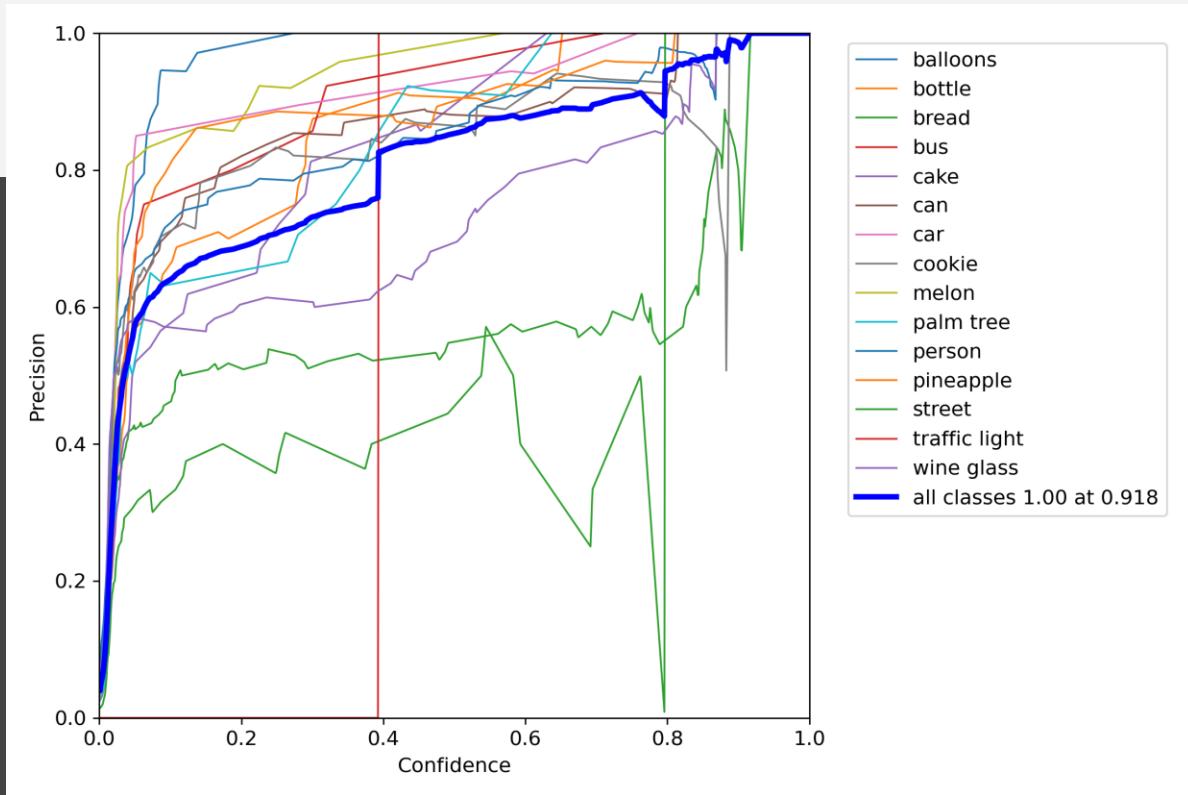


첫번째 그림은 매겨진 라벨값의 개수를 막대 차트로 나타낸 표이다.

두번째 그림은 라벨의 위치 좌표가 어디에 중점적으로 위치가 되어 있는지 알려주는 차트이다.

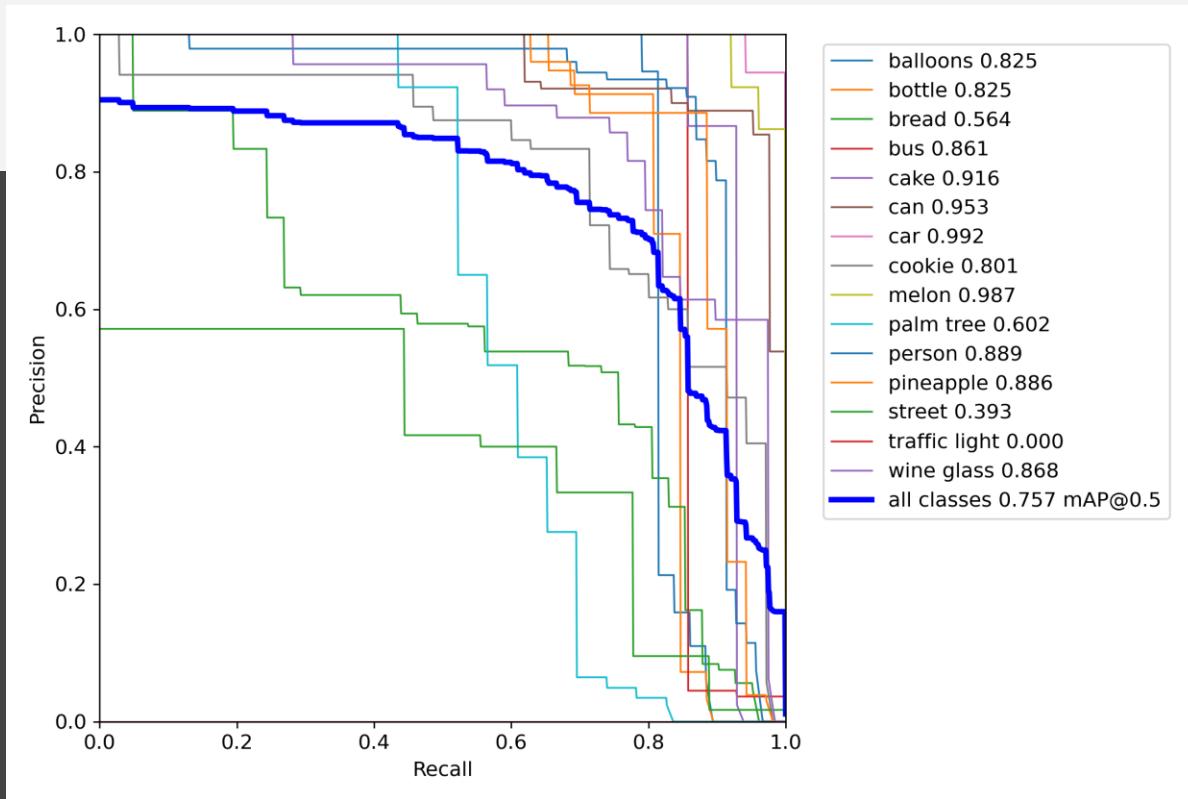
세번째 그림은 라벨의 가로와 세로의 크기값이 어디에 많이 위치 되어 있는지를 알려주는 차트입니다.

학습시킨 Yolov5 P_curve



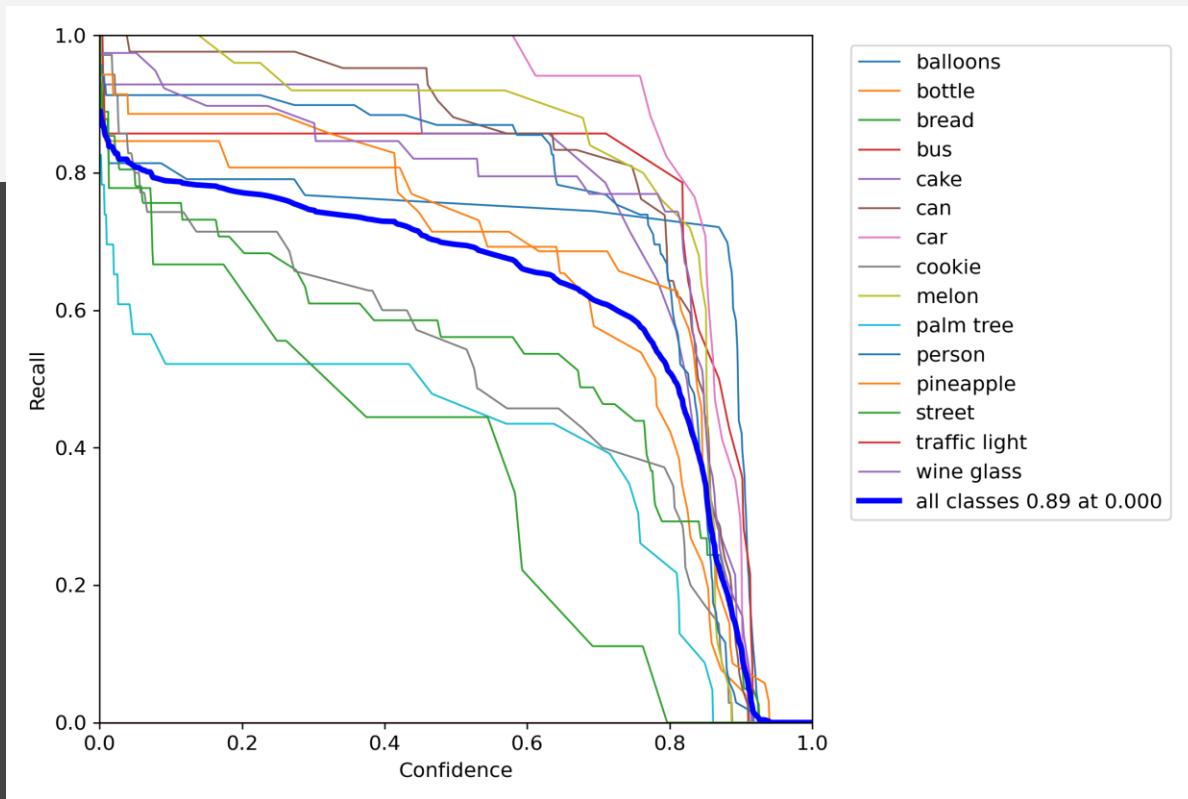
Precision(정밀도)와
Confidence(신뢰구간)을 이용한 그래프

학습시킨 Yolov5 PR_curve



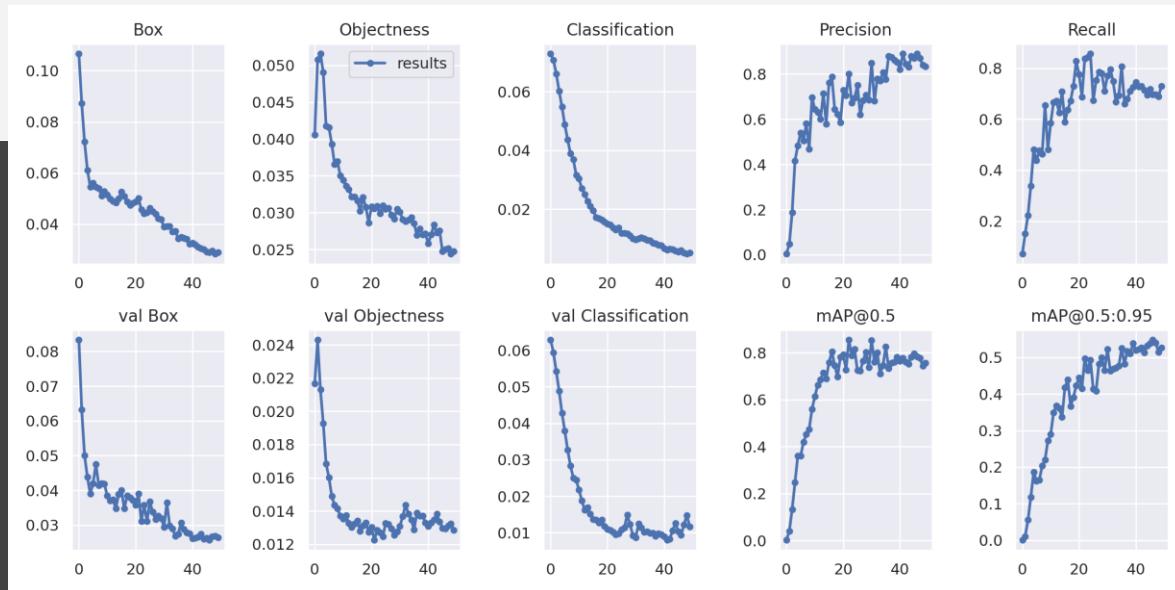
물체를 검출하는 알고리즘의 성능을 평가하는 방법 중 하나이다.
본 프로젝트로 예를 들면 분석을 통해 사물이나 물체가 얼마나 잘 인식 되었는지를 알려주는 차트입니다.

학습시킨 Yolov5 R_curve



Recall(검출율)과 Confidence(신뢰구간)을
가지고 나타낸 그래프입니다.

학습시킨 Yolov5 results



Box, val Box : Box의 의미는 물체를 인식하기위해 라벨링된 박스를 의미한다.
Objectness, Val Objectness : 기준 신뢰도
Classification, Val Classification : 분류
Precision : 옳게 검출한 비율
Recall : 실제 옳게 검출된 결과물 중에서 옳다고 예측한 것의 비율
Object Detection 의 성능 측정에는 mAP 를 사용한다.
각 클래스에 대해서 위의 방법으로 클래스별 AP(Average Precision) 를 구하고, 평균(Mean)한 것이 mAP 이다.
mAP@0.5:0.95 : 평균정밀도의 값이 0.5~0.95사이의 값들
mAP@0.5 : 평균정밀도의 값이 0.5이상인 값들



03

인스타그램
해시태그
크롤링

Selenium이란?

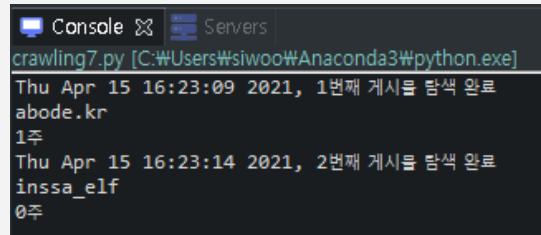
selenium은 주로 웹 앱을 테스트하는 웹 프레임워크입니다.

또한 webdriver의 API를 통해 브라우저를 제어하기 때문에 자바스크립트에 의해 동적으로 생성되는 사이트의 데이터를 크롤링할 때 매우 유용하게 사용되는 스크래핑 도구입니다.

[셀리니움의 장점]

1. 실제 브라우저를 실행시켜 동작하기 때문에 정적, 동적 페이지 모두 크롤링할 수 있다.
2. 웹사이트가 프로그램의 접근을 허용하지 않아도 크롤링이 가능하다.
3. 디버깅 시, 브라우저에서 눈으로 확인이 가능해 크롤링 과정을 확인할 수 있다.

자동 제어 되는 브라우저



```
Console > Servers
crawling7.py [C:\Users\siwoo\Anaconda3\python.exe]
Thu Apr 15 16:23:09 2021, 1번째 게시물 탐색 완료
abode.kr
1주
Thu Apr 15 16:23:14 2021, 2번째 게시물 탐색 완료
inssa_elf
0주
```

[셀리니움의 단점]

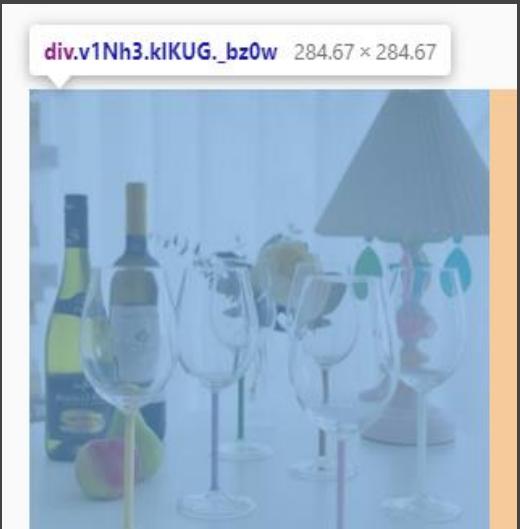
1. 브라우저를 실행시켜 동작하는 것 때문에 속도가 느리다.



Selenium을 이용해 크롤링

1. 사이트 분석하기

크롤링 전에 로그인이 필요한지, 검색의 구조는 어떻게 되어있는지 확인한다. 로그인이 필요한 경우 자동 로그인할 수 있게 코드를 작성해준다. 크롤링 하고 싶은 사이트의 구조를 정확하게 분석해 각 위치의 클래스명을 코드에 입력한다.



로그인 및 크롤링 시작 코드

```
# 해시태그 검색어
keyword = '검색어 입력'
count = 1000 # 몇 개의 글을 크롤링할지 입력

# 사전 정보 정의
username = 'ID 입력'
userpw = '비밀번호 입력'
time.sleep(3)

# 인스타 로그인 URL
loginUrl = 'https://www.instagram.com/accounts/login/'

# 해시태그 url 값
url = "https://www.instagram.com/explore/tags/{}/".format(keyword)

# Chrome driver 실행
driver = wd.Chrome(executable_path=r"chromedriver.exe")
driver.get(loginUrl)
time.sleep(2)

# login
driver.find_element_by_name('username').send_keys(username)
driver.find_element_by_name('password').send_keys(userpw)
time.sleep(2)
driver.find_element_by_css_selector('button.sqdOP.L3NKy.y3zKF').click()
time.sleep(3)

# 해시태그 검색창에 "검색어" 검색
driver.get(url)
time.sleep(10)

# 맨 왼쪽 상단 첫 게시글 클릭
driver.find_element_by_css_selector('div.v1Nh3.kIKUG._bz0w').click()
time.sleep(5)
```

데이터 크롤링 하여 파일로 저장

1. 사이트 분석을 통해 필요한 데이터의 위치 확인 확인 후 코드로 해당 위치의 데이터를 가져오게 한다.

2. 가져온 데이터를 리스트에 저장한다.

3. 오류로 인해 크롤링이 되지 않으면 50초 기다리고 다음 글로 이동된다.

4. 리스트에 있던 태그를 DF로 변경하여 CSV로 저장한다.

5. 모든 크롤링이 끝나면 크롬 드라이버를 자동으로 종료된다.

데이터 크롤링하여 파일로 저장되는 코드

```
# 데이터 기록, 다음 게시글로 클릭
for i in range(count):
    try:
        # 해쉬태그 데이터 기록
        data = driver.find_element_by_css_selector('.C7I1f.X7jCj')
        tag_raw = data.text
        tag = re.findall('#[A-Za-z0-9가-힣]+', tag_raw)
        tag = ''.join(tag).replace("#", " ") # "#" 제거
        tag_data = tag.split()
        for j in tag_data:
            instagram_tags.append(j)
    except:
        tag_data = "error"
        date_text = "error"

    try: # 최대 50초까지 기다렸다가, > 모양 클릭하여 다음 게시글로 넘어가기
        WebDriverWait(driver,50).until(EC.presence_of_element_located((By.CSS_SELECTOR, 'a._65Bje.coreSpriteRightPaginationArrow')).click()
    except:
        driver.close()

    time.sleep(5)
    print('{}, {}번째 게시글 흡식 완료'.format(time.strftime('%c', time.localtime(time.time())), i+1))

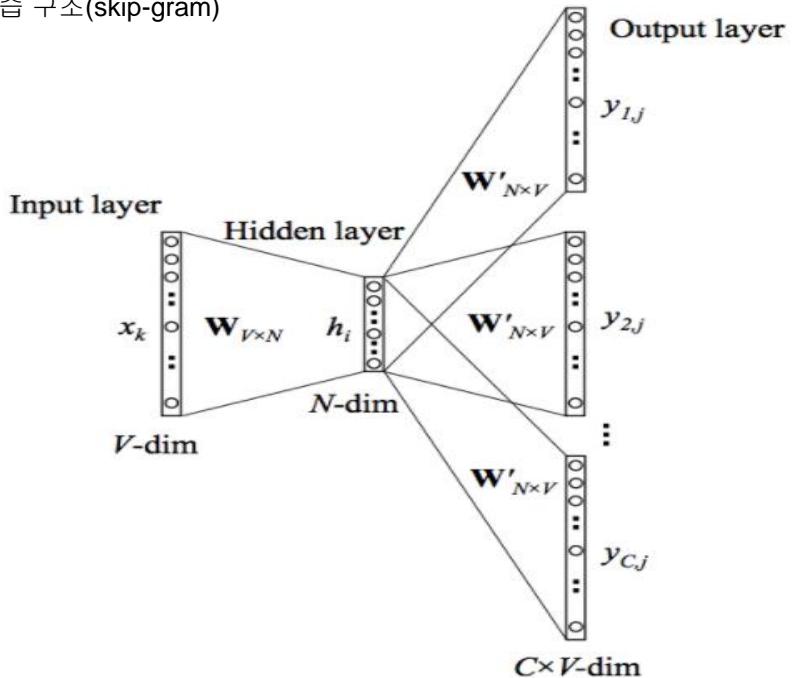
# 결과값 저장
insta_df = pd.DataFrame(instagram_tags)
insta_df.to_csv(keyword+'_'+results+'.csv', index = False, encoding='utf-8-sig')
# 크롬드라이버 종료
print('크롤링 종료')
driver.quit()
```

Word2Vec 키워드 도출

04

Word2Vec란?

학습 구조(skip-gram)



그림에서 입력층-은닉층, 은닉층-출력층을 이어주는 \mathbf{W} 와 \mathbf{W}' 는 가중치 행렬로써 서로 전치행렬입니다.

Word2Vec는 중심단어로 주변단어를 맞추거나, 주변단어로 중심단어를 더 잘 맞추기 위해 이 두개의 가중치 행렬들을 업데이트하면서 학습이 이루어지는 구조입니다.

가중치행렬(\mathbf{W})이 one-hot-encoding된 입력벡터와 은닉층을 이어주는 가중치 행렬임과 동시에 Word2Vec의 최종 결과물인 임베딩 단어벡터의 모임이기도 합니다.

출처 :

<https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/03/30/word2vec/>

Word2Vec 단어 유사도 코드

```
def Myword2vec(result):
    import collections
    counts = collections.Counter(result)      # 크롤링한 태그단어를 빈도수를 구하기.
    keyword = counts.most_common(1)[0][0]        # collections 의 most_common을 이용하여 가장 높은 빈도를 가진 단어 추출.

    fileName = '{}.txt'.format(keyword)
    with open(fileName, mode='w', encoding='utf_8') as fw:
        fw.write('\n'.join(result))

    genObj = word2vec.LineSentence(fileName)

    # Word Embedding(단어를 수치화)의 일종으로 word2vec
    model = word2vec.Word2Vec(genObj, size = 100, window=10, min_count=3, sg=1)
    # ** word2vec 옵션 정리 ***
    # size : 벡터의 크기를 설정
    # window : 고려할 앞뒤 풀 ( 앞 뒤의 단어를 설정 )
    # min_count : 사용할 단어의 최소빈도 ( 설정된 빈도 이하 단어는 무시 )
    # workers : 동시에 처리할 작업의 수
    # sg : 0 (CBOW) , 1 (Skip-gram)
    # CBOW 는 주변 단어를 통해 주어진 단어를 예측하는 방법
    # Skip-gram 은 하나의 단어에서 여러 단어를 예측하는 방법
    # 보편적으로 Skip-gram이 많이 쓰인다.
    model.init_sims(replace=True)    # 필요없는 메모리 해제

    # 모델 저장
    try:
        model.save('C:/work/psou/final_project/myapp/static/model/' + keyword + '.model')
    except Exception as e:
        print('err : ', e)

    # 모델 불러오기
    model = word2vec.Word2Vec.load('C:/work/psou/final_project/myapp/static/model/' + keyword + '.model')

    # 사진 속의 단어와 가장 유사한 단어 추출 및 DataFrame으로 저장
    result = pd.DataFrame(model.wv.most_similar(keyword, topn = 10), columns=[['단어'], ['유사도']])
```

Word2Vec 시각화 코드

```
# 시각화1 (유사도 구하기)
sns.set(style='whitegrid', font='Malgun Gothic', font_scale=1)
graph = sns.PairGrid(result.sort_values('유사도', ascending=False), y_vars=['단어']) # 유사도가 높은것부터 내림차순으로 정렬.
graph.fig.set_size_inches(5, 10)
graph.map(sns.stripplot, size = 10, orient = 'h', palette = 'ch:s=1, r=-1, h=1_r', linewidth = 1, edgecolor='w')
graph.set(xlim = (0.0, 1), xlabel = '유사도', ylabel = '')
titles = keyword
# 자트 제목과 가로축을 생성.
for ax, title in zip(graph.axes.flat, titles):
    ax.set(title = titles)
    ax.xaxis.grid(False); ax.yaxis.grid(True)    # 수직격자 False, 수평격자 True

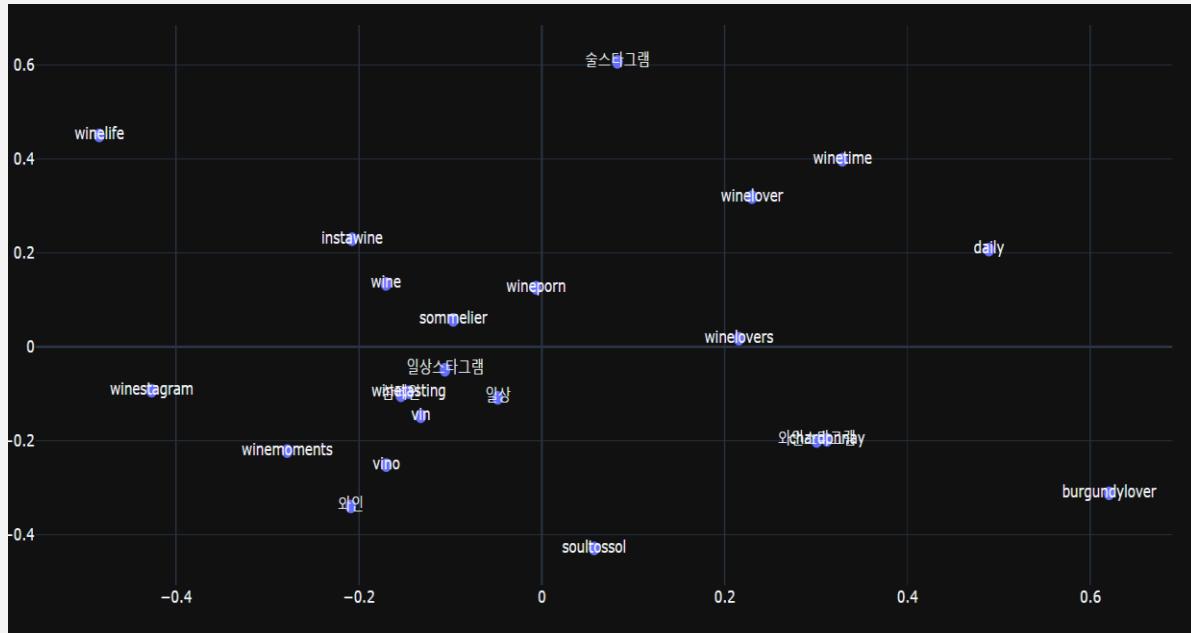
    sns.despine(left = True, bottom = True)
plt.savefig('C:/work/psou/final_project/myapp/static/word2vecimg/{}1.png'.format(keyword), bbox_inches='tight') # 경로설정
plt.close()

# 시각화 2 (모델의 산점도 구하기 - PCA 이용)
word_vectors = model.wv
vocab = word_vectors.wv.vocab # 벡터화 시킨 모델의 단어들을 추출. gensim 3version에서는 wv.vocab, gensim 4version에서는 wv.index_to_key를 사용.
vocab = list(vocab.keys())
word_vectors_list = [word_vectors[v] for v in vocab] # 해당 단어들을 list로 저장.

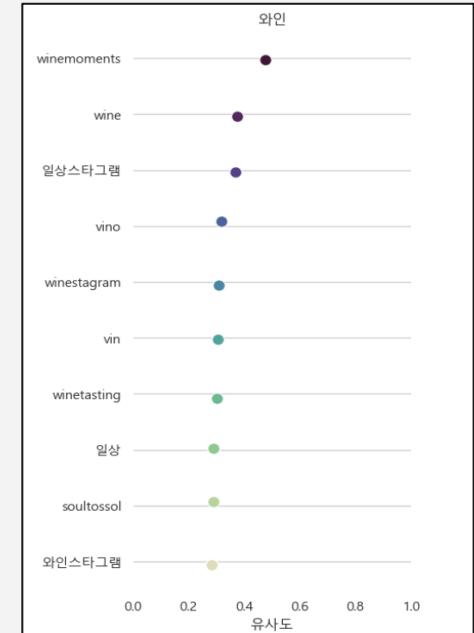
pca = PCA(n_components = 2) # PCA 사용.
xys = pca.fit_transform(word_vectors_list)
xs = xys[:,0]
ys = xys[:,1]
s = 80
plt.scatter(xs, ys, marker='o', alpha = 0.7, edgecolors = 'w' , cmap='Green', s = s)
for i, v in enumerate(vocab):
    plt.annotate(v, xy=(xs[i], ys[i]))
plt.grid(False)
plt.savefig('C:/work/psou/final_project/myapp/static/word2vecimg/{}2.png'.format(keyword), bbox_inches='tight') # 경로설정
plt.close()

# 시각화3 (모델의 산점도 구하기. html 형식으로 저장하여 동적으로 산점도를 들여다 볼 수 있음.)
# pip install plotly - plotly를 이용하기 위한 패키지를 anaconda 프롬프트에서 실행.
fig = go.Figure(data=go.Scatter(x = xs, y = ys, mode='markers+text', text=vocab, marker=dict(size=80)))
fig.update_layout(template='plotly_dark')
fig.update_layout(title=keyword)
po.write_html(fig, file = 'C:/work/psou/final_project/myapp/static/word2vecimg/{}3.html') # 경로설정
```

모델의 산점도(PCA)



유사도



학습시킨 모델에 중심단어(와인)을 넣었을 때 나온 유사도 그래프입니다. 산점도에서 '와인'과 가까운 단어순으로 나오는 것을 볼 수 있습니다. 즉, 중심단어를 '와인'이 아닌 'winemoments'로 선택했을 때도 '와인'이란 단어는 유사도가 높게 나옵니다.

05

LDA
키워드 도출

토픽모델링 – 잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)이란?

각 단어나 문서의 숨겨진 주제를 찾아내어 문서와 키워드 별로 주제끼리 묶어주는 비지도 학습 알고리즘입니다.

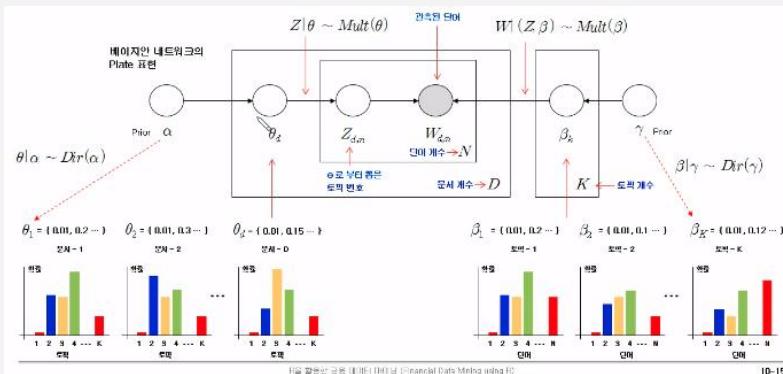
Latent는 사전적인 의미는 “잠재적인, 숨어 있는” 입니다. LDA에서는 관측 가능한 단어 들로부터 알 수 없는(Latent) 문헌 별 주제분포(θ), 주제별 단어분포(ϕ)를 추론하는 작업을 뜻합니다.

Dirichlet는 문헌별 주제분포와 주제별 단어분포는 디리클레 분포를 따른다고 가정한다는 뜻입니다.

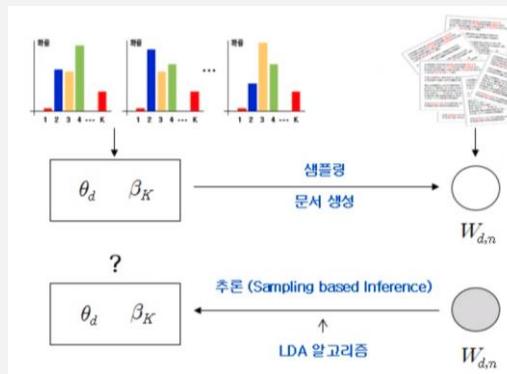
Allocation은 사전적인 의미는 “할당”입니다. LDA에서는 각 단어를 결정할 때, 문헌 별 주제분포에 대한 다항 분포(Multinomial Distribution)로 주제를 ‘할당’ 한 뒤 그 주제로부터 단어 추출합니다.

※ 디리클레분포(Dirichlet)는 이항분포가 아닌, 연속확률 분포 중 하나로 k 차원의 실수 벡터 중 벡터의 요소가 양수이며 모든 요소를 더한 값을 1로 하여 확률 값이 정의되는 분포입니다.

<LDA가 가정하는 문서생성 과정 이미지>



〈관측단어로 잠재 변수 추론하여 토픽 모델링한 이미지〉

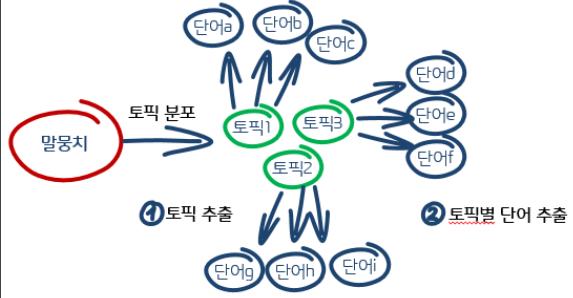


LDA 원리란?

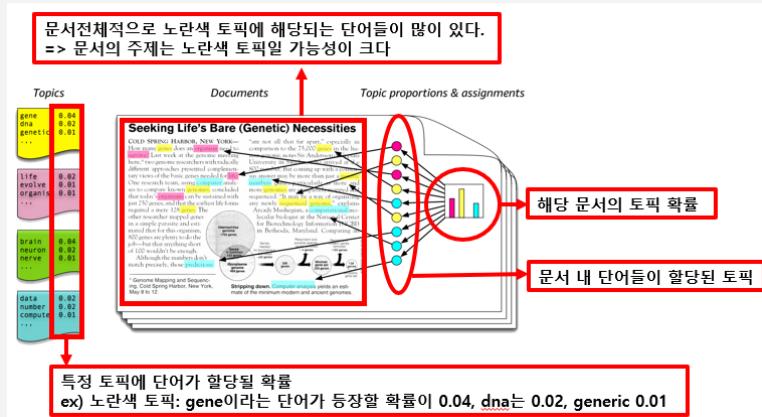
디리클레(Dirichlet)의 분포를 가정하고 토픽이라는 잠재(Latent)된 변수를 활용하여 각 문서들과 단어들이 토픽에 할당되는 확률 분포를 그린다.

[LDA가 토픽에 적절한 단어를 할당하는 과정]

1. 각 문서 내 문장들 토큰화, 불용어 처리를 한다.
2. 토픽의 개수 지정을 한다.
3. 단어들을 랜덤하게 토픽들에 할당을 해준다.
4. A라는 단어를 제외하고는 토픽에 맞게 할당되었다고 할 때, A라는 단어를 A라는 단어를 어떤 토픽에 적절한지를 추정 후 그 토픽에 할당 한다.
5. 나머지 단어들도 하나씩 4번의 과정을 반복하여 최종적으로 문서 전체를 정해진 개수만큼의 토픽들로 분류한다.



예시1



예시2

“Arts”	“Budgets”	“Children”	“Education”
NEW	MILLION	CHILDREN	SCHOOL
ITEM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BUDGET	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HOURS
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTING	NOTES	SAVING	DEPARTMENT
FIRST	STATE	FAMILY	MANAGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONET	MEN	STATE
THEATER	PROGRAMS	GOVERNMENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give \$1.25 million to Lincoln Center, Metropolitan Opera Co., New York City Ballet, and Juilliard School. "Our arts community felt that we had a real opportunity to make a mark on the future of the performing arts with these grants, an act of generosity as we traditional areas of giving in health, medical, education, and the social services. Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be \$200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York City Ballet will each get \$300,000. The Juilliard School, where music and the performing arts are taught, will get \$300,000. The Hearsts, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual \$100,000 donation, too.

LDA의 모델 도표

Reference: "Latent Dirichlet Allocation", Journal of Machine Learning Research, 2003.

모델 평가기준

LDA의 토픽 수는 사용자가 지정하는 하이퍼 파라미터이다. 최적 토픽 수 또한 여러 실험을 통해 구해야 하는 미지수이며, 보편적으로 최적 토픽 수를 구하는 데 쓰는 Perplexity와 Perplexity의 한계를 극복하기 위해 제시된 Topic Coherence지표가 존재합니다.

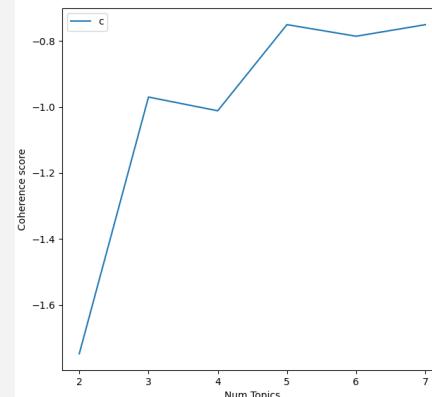
Perplexity

- 사전적 의미로 "당혹, 혼란, 곤혹" 등의 의미
- 확률모델이 실제로 관측되는 값을 얼마나 잘 예측하는지를 평가할 때 사용
- perplexity 값이 작을수록 해당 토픽 모델은 실제 문헌 결과를 잘 반영한다는 뜻이므로 학습이 잘 되었다고 평가
- 주 용도
 - 동일 모델 내 파라미터에 따른 성능 평가할 때 주로 사용
 - 선정된 토픽 개수마다 학습시켜 가장 낮은 값을 보이는 구간을 찾아 최적화된 토픽의 개수 선정
- 한계
 - Perplexity가 낮다고 해서, 결과가 해석 용이하다는 의미가 아님
 - 낮은 Perplexity를 갖는 모델은 학습이 잘되었다는 뜻일 뿐 그 결과가 사람이 해석하기에 좋다는 것이 아님

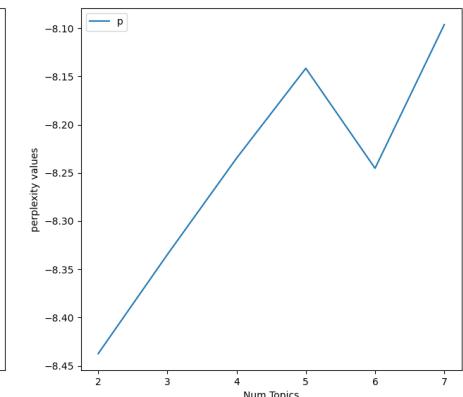
Coherence

- 토픽이 얼마나 의미론적으로 일관성 있는지 판단할 때 사용
- 실제로 사람이 해석하기에 적합한 평가 척도
- Coherence 값이 높을수록 토픽 내의 단어 간의 유사성이 높다고 평가
- 주 용도 : 해당 모델이 얼마나 실제로 의미 있는 결과를 내는지 확인
- 평가를 진행하기 위해 다른 외부 데이터(코퍼스, 시소러스 등)가 필요

<단어간의 유사성을 수치 표현>



<모델 정확도 예측 수치 표현>



[데이터 처리]

1. 인스타그램 크롤링 후 단어들을 저장한 csv파일을 pandas 함수로 불러옴

한글만 추출, 결측치 제거

2. 이미 단어들로만 구성된 파일이기 때문에 각 문서의 문장들을 토큰화한 것과 같은 데이터로 본다

[gensim을 통해 Corpus(말뭉치) Dictionary(사전) 언어모델 형성]

1. LDA 적용을 위한 텍스트의 벡터화

2. 말뭉치 생성

Corpus (66, 2): 정수 인코딩이 66으로 할당된 단어가 특정 문서에
두 번 등장함을 의미.

Dictionary를 이용하여 66번 째의 단어를 확인

```
# 1. 데이터 처리
tokenized_doc = []
for i in range(4):
    tok_list = pd.read_csv('c:/work/psou/final_project/myapp/static/crawling/'+str(i)+'_results.csv')['e'].\
        str.extract('([ㄱ-ㅎㅏ-ㅣㅏ-ㅣㄱ-ㅎ]+)', expand=False).dropna().tolist()
    tokenized_doc.append(tok_list)

# 2. gensim을 통해 Corpus(말뭉치) Dictionary(사전) 언어모델 형성
dictionary = corpora.Dictionary(tokenized_doc)
corpus = [dictionary.doc2bow(text) for text in tokenized_doc]
```

[적절한 토픽 수를 결정하기 위해 Perplexity 및 Coherence 계산]

1. LDA모델은 사용자가 직접 토픽의
개수를 지정해야 한다.

2. 토픽 개수에 따라 Perplexity와
Coherence의 변화를 보고 Perplexity가
가장 낮은 토픽의 수로 모델 학습 진행

3. gensim의 CoherenceModel의
get_coherence 함수를 사용하여
Coherence 계산

4. gensim의 ladmodel의 log_perplexity
함수를 사용하여 Perplexity 계산

```
# 3. 적절한 토픽수를 결정하기 위해 Perplexity 및 Coherence 계산
# 3-1. Perplexity 및 Coherence 계산 함수 정의
def compute_metrics_values(dictionary, corpus, limit, start=2, step=3):
    coherence_values = []
    perplexity_values = []
    model_list = []
    for num_topics in range(start, limit, step):
        model = gensim.models.LdaModel(corpus, num_topics = num_topics, id2word=dictionary, passes=15)
        model_list.append(model)
        coherence_model_lda = gensim.models.CoherenceModel(model=model, \
            corpus=corpus, dictionary=dictionary, coherence='u_mass')
        coherence_values.append(coherence_model_lda.get_coherence())
        perplexity_values.append(model.log_perplexity(corpus))
    return model_list, coherence_values, perplexity_values

# 3-2. compute_metrics_values 함수 호출
model_list, coherence_values, perplexity_values = \
    compute_metrics_values(dictionary=dictionary, corpus=corpus, start=2, limit=8, step=1)

# 3-3. Perplexity 및 Coherence 계산 결과 출력
limit=8; start=2; step=1;
x = range(start, limit, step)

dic = {}
for m, cv in zip(x, coherence_values):
    print("Num Topics = ", m, " has Coherence Value of", round(cv, 4))

for m, pv in zip(x, perplexity_values):
    print("Num Topics = ", m, " has perplexity values of", round(pv, 4))

dic[abs(round(pv, 4))] = m

# 4. Perplexity가 가장 낮은 토픽수 지정
a = min(dic.keys())
```

<토픽 개수에 따른 Perplexity 및 Coherence 점수>

Num Topics = 2	has Coherence Value of -1.3358
Num Topics = 3	has Coherence Value of -1.4074
Num Topics = 4	has Coherence Value of -0.7819
Num Topics = 5	has Coherence Value of -0.4707
Num Topics = 6	has Coherence Value of -0.6225
Num Topics = 7	has Coherence Value of -0.6536
Num Topics = 2	has perplexity values of -8.3444
Num Topics = 3	has perplexity values of -8.2564
Num Topics = 4	has perplexity values of -8.1324
Num Topics = 5	has perplexity values of -8.0802
Num Topics = 6	has perplexity values of -8.0866
Num Topics = 7	has perplexity values of -8.0965

```

# 5. LDA 모델
# 5-1. LDA모델 생성
file = 'C:/work/psou/final_project/myapp/static/model/lda.model'
NUM_TOPICS = dic[a] # dic[a]개의 토픽
lda = gensim.models.LdaModel(corpus, num_topics = NUM_TOPICS, id2word=dictionary, passes=15, random_state=121)

# 5-2. LDA모델 저장
lda.save('C:/work/psou/final_project/myapp/static/model/lda.model')

# 5-3. LDA모델 불러오기
fname = datapath("C:/work/psou/final_project/myapp/static/model/lda.model")
lda = gensim.models.LdaModel.load(fname, mmap='r')

# 6. 학습시킨 모델의 각 토픽에 할당된 단어 리스트
word = gensim.models.coherencemodel.CoherenceModel.top_topics_as_word_lists(lda, dictionary, topn=30)

# 7. 토픽 인덱스와 각 토픽에 할당된 단어들의 가중치
topics = lda.print_topics(num_words=30)
for topic in topics:
    print(topic)

# 8. 학습한 모델에 대한 평가 지표 Perplexity 및 Coherence
# 8-1. Perplexity Score: a measure of how good the model is. lower the better.
print('\nPerplexity: ', lda.log_perplexity(corpus))

# 8-2. Coherence Score
coherence_model_lda = gensim.models.CoherenceModel(model=lda, corpus=corpus, dictionary=dictionary, coherence='u_mass')
coherence_lda = coherence_model_lda.get_coherence()
print('Coherence Score: ', coherence_lda)

```

[LDA 모델]

-num_topics: 지정할 토픽을 수
 -Passes: 알고리즘의 동작 횟수
 -알고리즘이 결정하는 토픽의 값이 적절히 수렴할 수 있도록
 충분히 적당한 횟수를 정해주면 됩니다. 여기서는 총 15회를
 수행하였습니다.



각 topic별 단어 리스트
 [['캔', '재활용', '플라스틱', '커피', '플로깅', '제주도', '수중정화', '제주바다', '플로깅', '프리다이빙', '해
 각 topic을 구성하는 단어와 가중치
 (0, '0.051*"캔" + 0.008*"재활용" + 0.007*"플라스틱" + 0.006*"커피" + 0.006*"플로깅" + 0.005*"제주도" -
 (1, '0.043*"와인잔" + 0.012*"와인" + 0.007*"커피잔" + 0.006*"홈카페컵" + 0.005*"고볼렛잔" + 0.005*"집들
 (2, '0.029*"빵" + 0.024*"케이크" + 0.010*"디저트" + 0.009*"빵스타그램" + 0.009*"스타그램" + 0.008*"?

Perplexity: -8.133149703189432

Coherence Score: -0.22485361643405244

[출력결과]

-각 단어 앞에 붙은 수치는 단어의
 해당 토픽에 대한 기여도를 보여줍니다.
 -또한 맨 앞에 있는 토픽 번호는 0부터
 시작하므로 총 3개의 토픽은 0부터
 2까지의 번호가 할당되어져 있습니다.

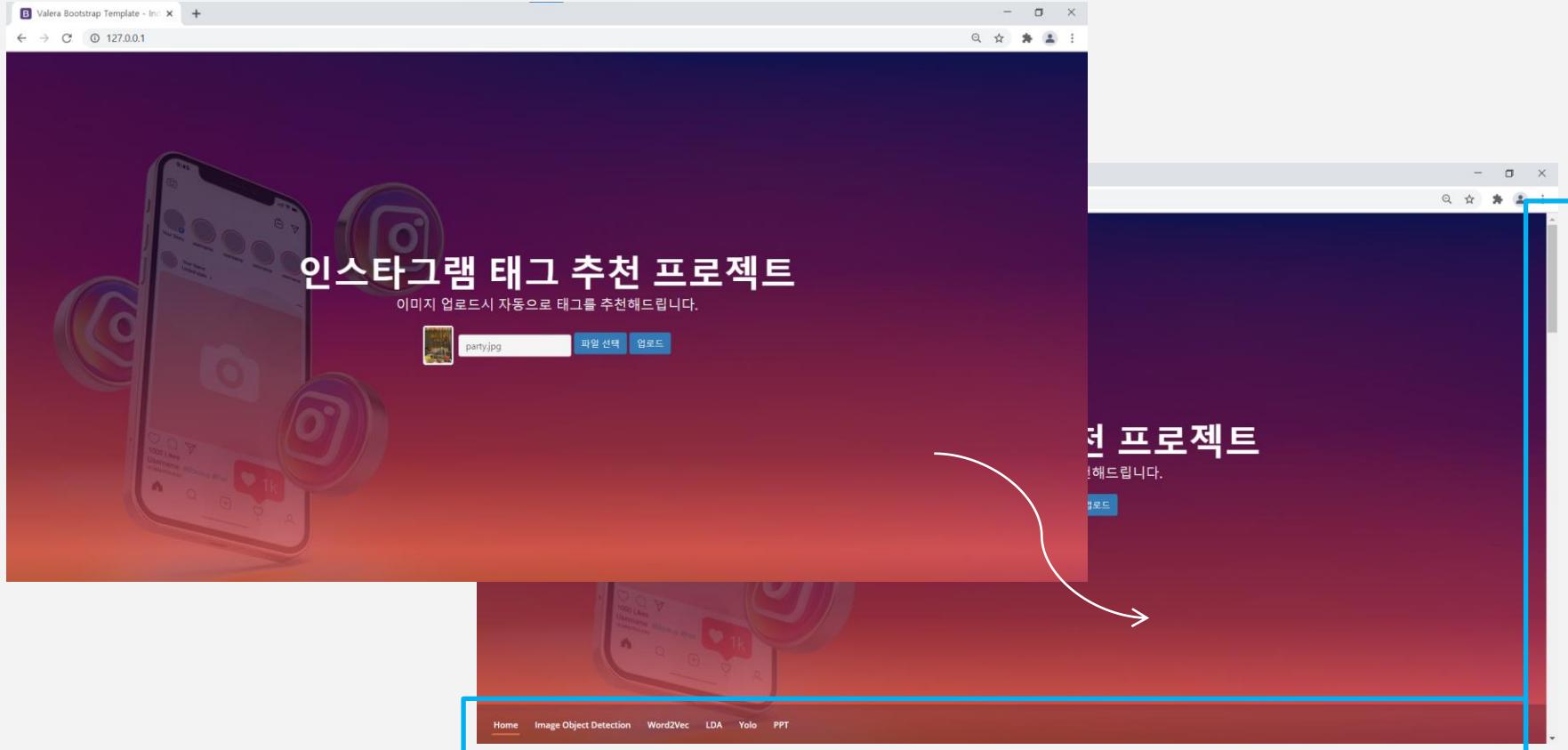


06

장고
웹 출력

[메인 화면]

1. 태그추천을 받고 싶은 이미지를 넣어줍니다.
2. 이미지를 업로드 하게 되면 메인페이지 하단에 메뉴가 생기면서 스크롤이 자동 생성 됩니다.



[Image Object Detection]

왼쪽에는 업로드된 이미지가 오른쪽화면에는 객체 인식된 이미지가 보여지는 것을 알 수 있습니다.

Image Object Detection show

이미지가 들어오면 한 부분에 물체가 있다는 걸 인식 (Object Recognition)하고, 그 물체가 무엇인지(Object Classification)판단하고 정확한 위치를 찍어 줍니다(Object Localization).
OC와 OL이 합쳐지면 Object Detection, OD 가 됩니다

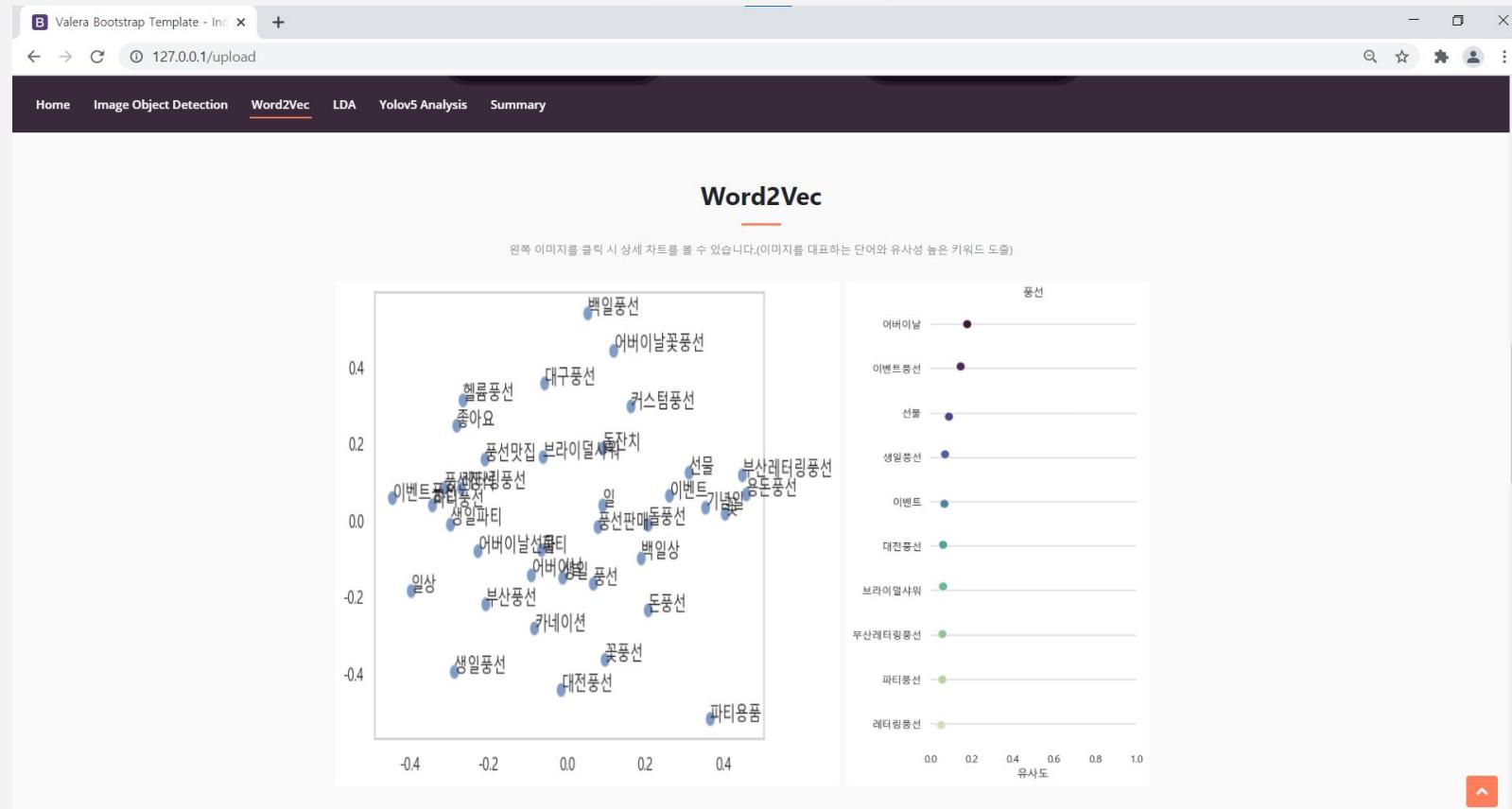
LOREM IPSUM
Your Location

LOREM IPSUM
Your Location

bread 0.45 balloon 0.96
wine glass 0.36 bacon 0.92
bread 0.26 bacon 0.88
bread 0.26 bacon 0.81

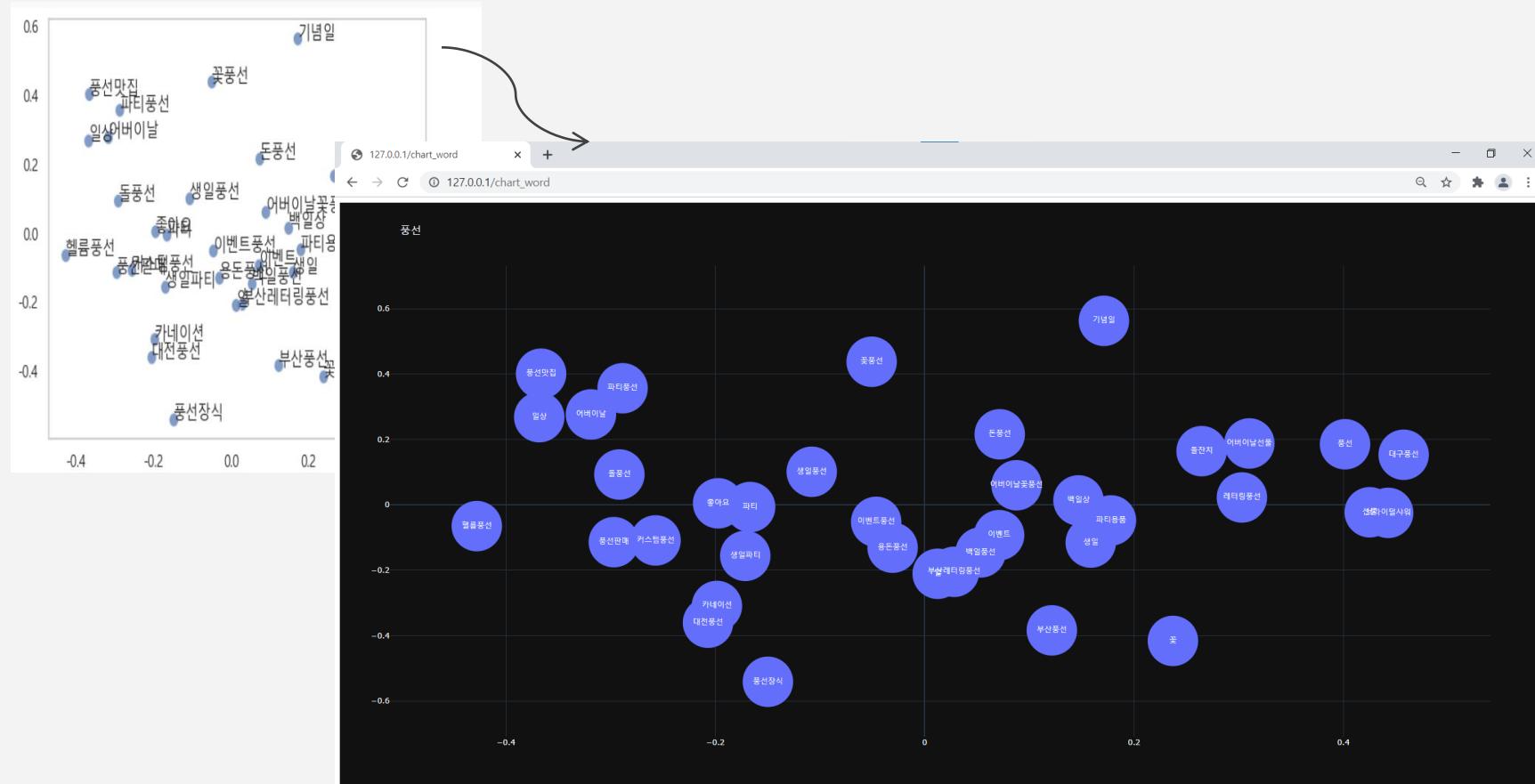
[Word2Vec]

이미지를 대표하는 단어 하나를 뽑아 유사성 높은 키워드를 도출하였습니다.



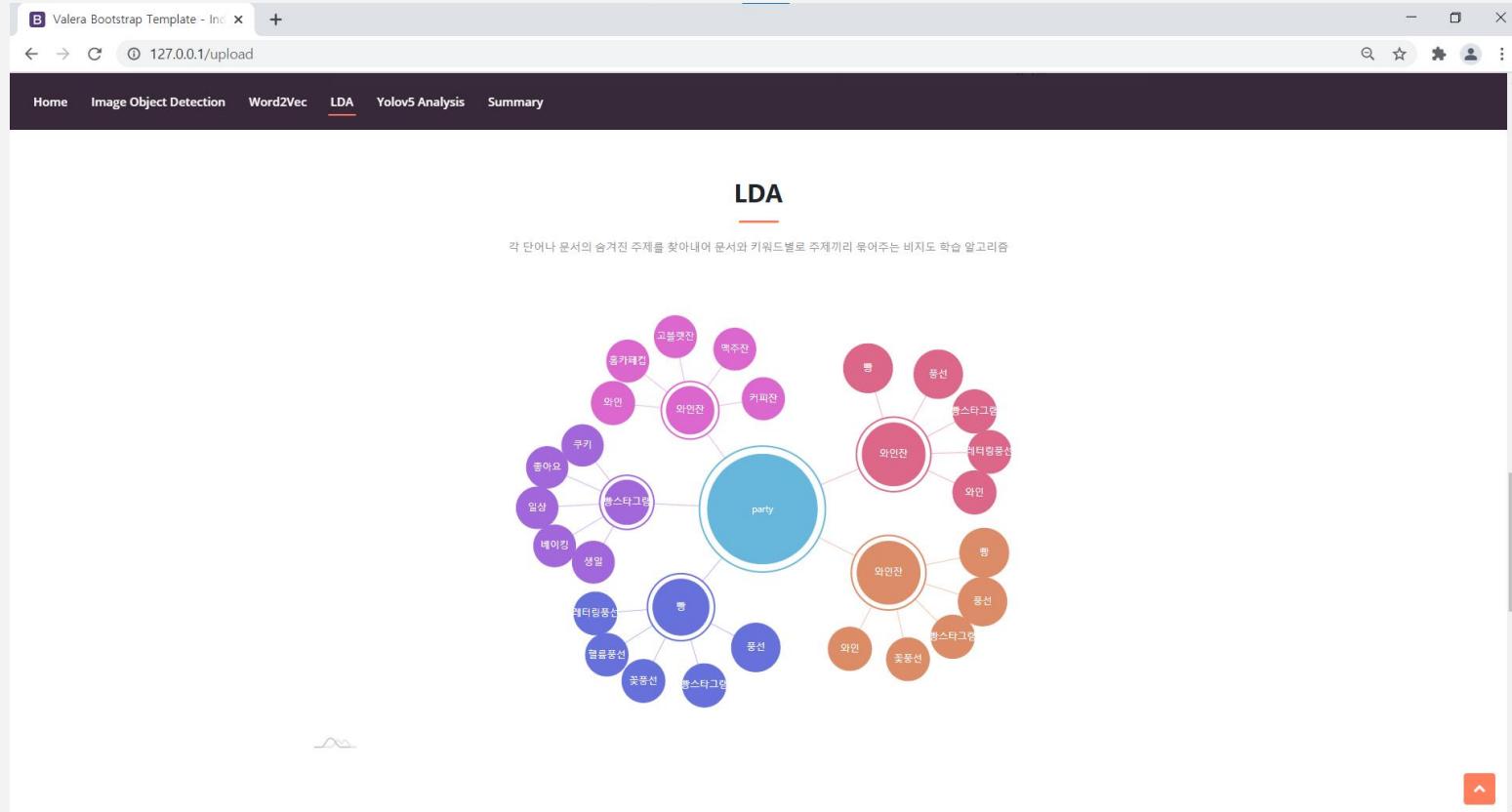
[Word2Vec]

페이지내에서 아래와 같은 이미지를 클릭시 상세한 차트를 볼 수 있습니다.



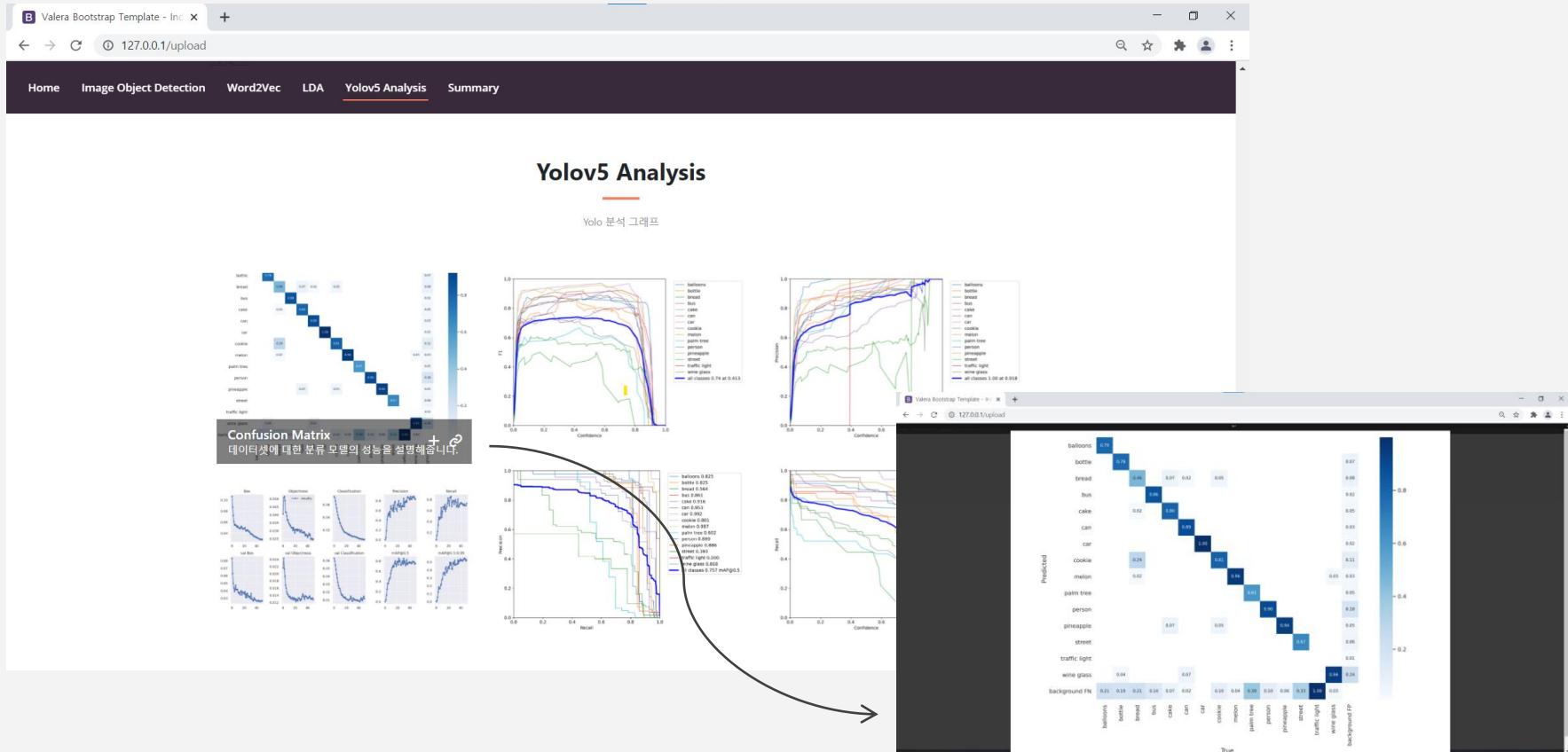
[LDA]

각 단어나 문서의 숨겨진 주제를 찾아내어 문서와 키워드별로 주제끼리 묶어주는 비지도 학습 알고리즘입니다.
아래와같이 키워드 차트가 도출되는 것을 볼 수 있습니다.



[Yolov5 Analysis]

Yolov5를 통해 학습시킨 데이터셋을 분석한 차트들이다.
이미지를 클릭하게 되면 이미지를 상세히 볼 수 있습니다



[Summary]

프로젝트에 대한 설명자료를 다운로드 가능합니다.

The image shows a screenshot of a web application interface. On the left, a large window displays a 'Summary' page with a background image of a mountain range at dusk. The page features the word 'Summary' in large white text and a smaller text message in Korean: '프로젝트에 대한 설명자료를 다운로드 가능합니다.' (Downloadable project description). A prominent orange '다운로드' (Download) button is centered. The top navigation bar includes links for Home, Image Object Detection, Word2Vec, LDA, Yolov5 Analysis, and Summary. The URL in the address bar is 127.0.0.1/upload. On the right, a separate window titled 'Investment Business Plan' is open, showing a PDF document with the same content as the summary page. The PDF viewer interface includes a navigation bar with icons for back, forward, and search, and a progress bar indicating 1 / 43 pages. A watermark for 'Valera Bootstrap Template - Inc' is visible in the top left corner of the PDF viewer.

프로젝트를 하면서 느낀점

최인권 : 다양한 모델에 대해서 한층 더 잘 알 수 있게 되었고 모델을 만들기 위해 전처리 과정이 굉장히 까다롭고 손이 많이 가는 작업이란걸 새삼 느끼게 되었습니다. 시각화하는 과정에서도 어떻게 효과적으로 전달 할 수 있을지 다양한 방법을 찾아보며 각각의 모델에 적합한 시각화 방법도 배우는 좋은 시간이었습니다.

이준형 : 장점은 프로젝트를 진행할때 커뮤니케이션이 중요하다는걸 깨달았다 각자 힘들때 아낌없이 격려를 해주는게 힘이된다는걸 알았다. 또 맑은 역할의 책임감이 증가하고 일처리가 좀 더 빨라졌던것 같다. 단점은 안배운걸 써보려고하니 남이 짜놓은 코드를 읽고 이해하는 시간이 많이들었다. 감사합니다 형님들 나중에 판교에서봐요 ❤

김다해 : 처음 접하는 객체 인식을 검색과 영상을 통해 공부하면서 코드를 구성했다. 하나씩 문제를 해결해나가자 전체 흐름이 풀리는것 같았다. 이번 프로젝트는 자바 프로젝트와 달리 코드를 구성하는 시간 보다 데이터 셋을 만들고 모델 전처리 하는 시간이 더 걸렸던 것 같다. 더 다양한 모델로 해보지 못한 아쉬움이 있었지만 프로젝트를 마치면서 많은 공부를 한 것 같았다. 처음 접하더라도 내것으로 만들며 끊임없이 노력하는 개발자가 되어야겠다고 생각했다. 마지막으로 팀원들 모두 고생하셨습니다.

이시우 : 개발자를 커리어를 전향하면서 다시 축준할 막막했었다. 그래도 학원을 다니면서 기초도 다지고 두 번의 프로젝트를 하면서 개발에 대해 많이 배운 것 같다. 초반에 강의가 비대면으로 바뀌면서 아쉬웠지만 과정을 잘 마무리할 수 있어서 다행이다. 파이널 프로젝트를 하면서 강의 중 배우지 않은 모델도 써봤는데 공부할게 정말 많다고 느꼈다. 수료 후 개발자가 된 뒤에도 공부를 꾸준히 해야겠다고 생각했다. 6달동안 잘 지도해주셔서 감사합니다 :)

조성준 : 파이썬을 배우면서 분석중에 이미지분석이 가장 신기했고 실제로 프로젝트에 적용시켜보니 배우던 함수들과는 다르게 엄청 어려웠습니다. 용어부터 분류하는 방법까지 정말 모르는게 많았지만 마지막까지 해낸것 같습니다. 진행하면서 팀원들이 열심히 찾아가며 어려운 산을 하나 넘은게 큰 도움이 되었을지 모르겠습니다. 저도 큰 도움이 많이 되었고 새로운 기술을 또 하나 알아간게 모두에게 도움되었으면 좋겠습니다.

서희진 : 프로젝트를 진행하면서 부족한 부분을 확실히 알게되었고 조금씩 채울 수 있게되었습니다. 공부했던 부분들을 확실히 다질 수 있었지만 앞으로 더 공부해야하는 부분들도 파악할 수 있었던 시간이었습니다 데이터 수집과정에서부터 많은 어려움이 있었지만 팀원들이 포기하지 않고 적극적으로 참여하고 각자 맡은 임무들을 잘 수행해줘서 잘 마무리할 수 있었습니다. 모든 팀원들에게 감사합니다.

감사합니다.