채용공고 & 기업리뷰 크롤링 프로젝트

Project Github address
https://github.com/juhyunson/Job_opening_Company_review

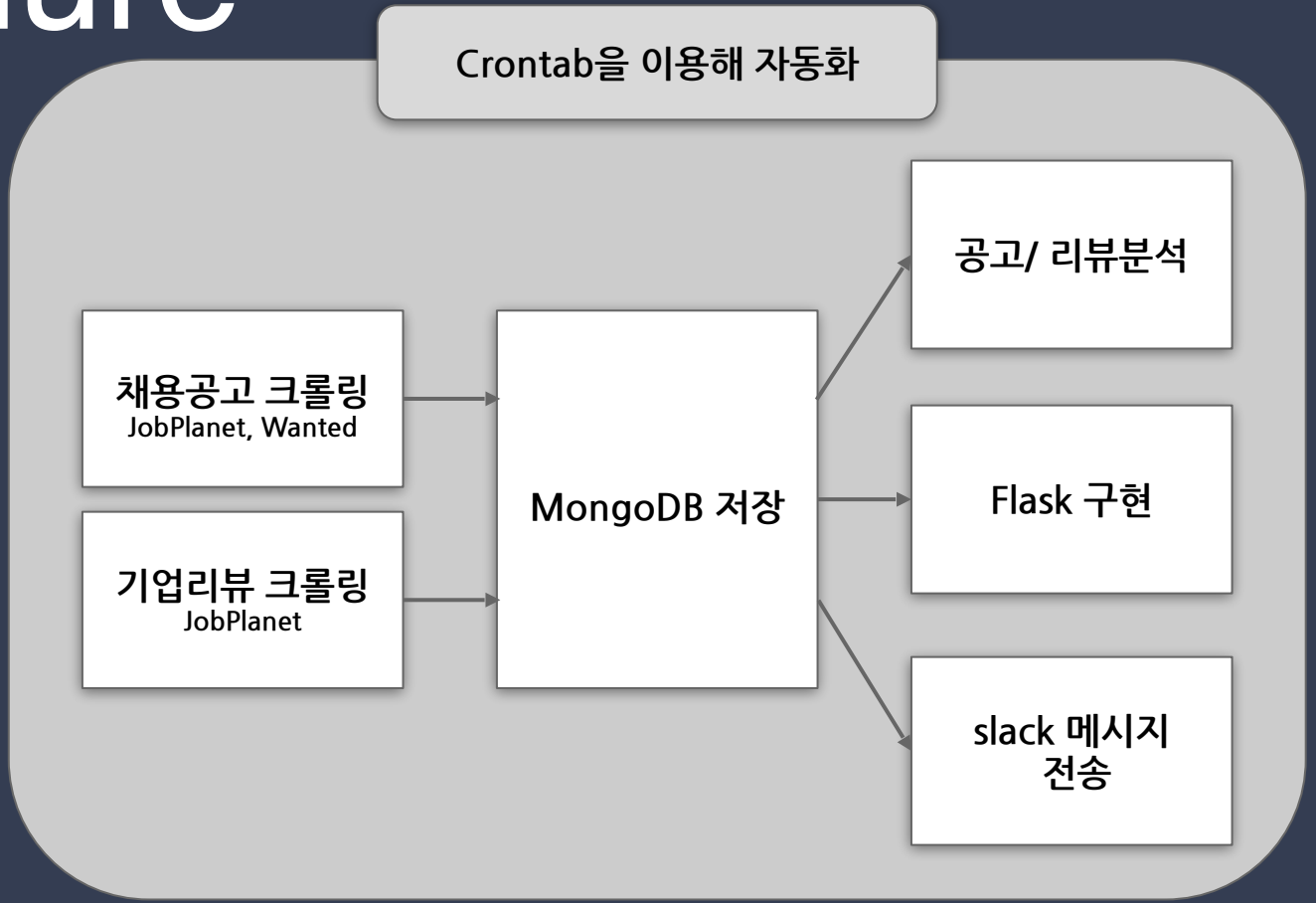
손주현
Github:https://github.com/juhyunson

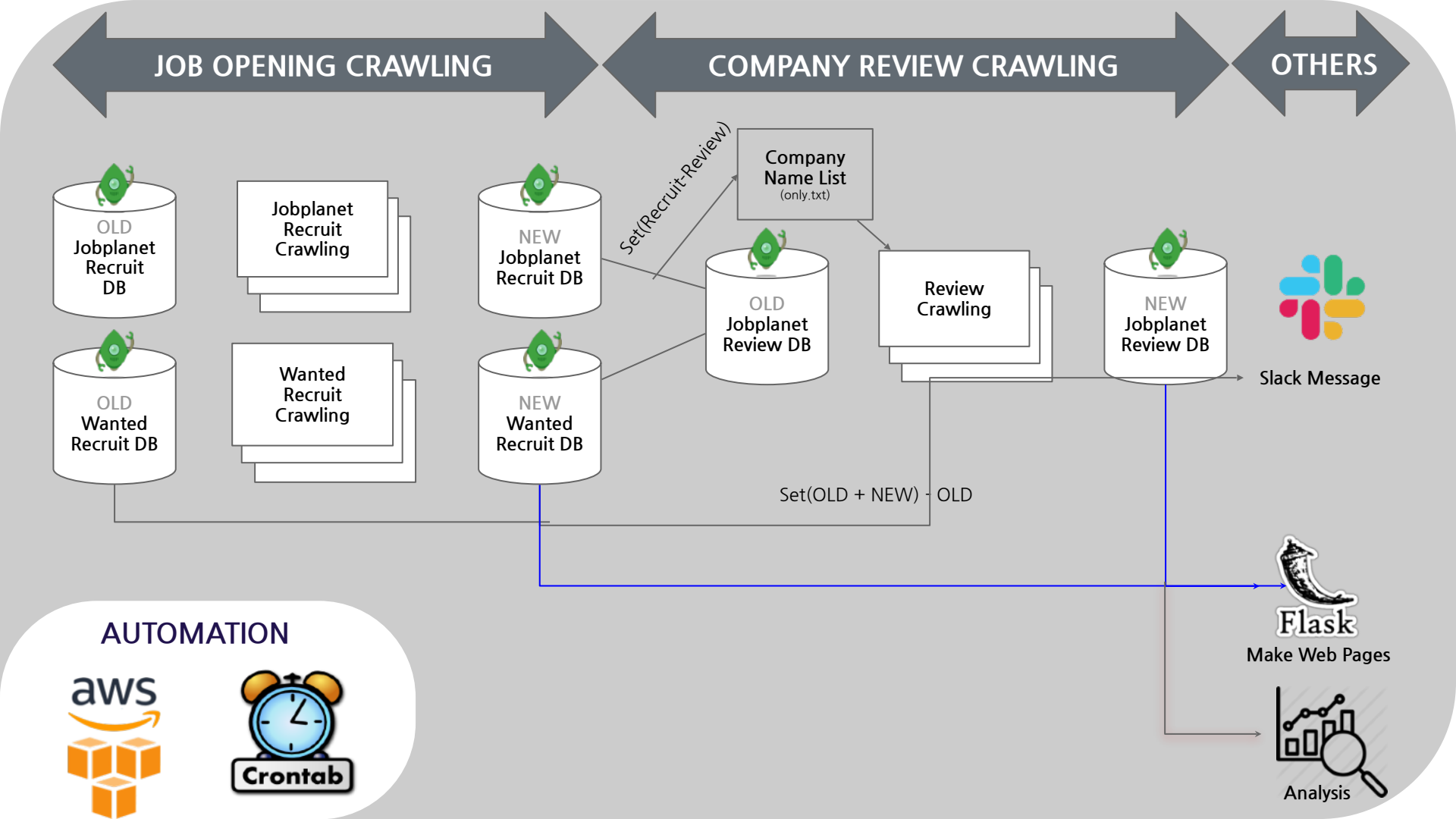
이진주
Github: https://github.com/pearl-lee

Purpose

데이터 사이언스 관련 직무로의 취업을 위해
채용공고와 그 공고를 게시한 기업의 리뷰를
매일 ! 한번에! 보고싶다! 알림도 받으면서!

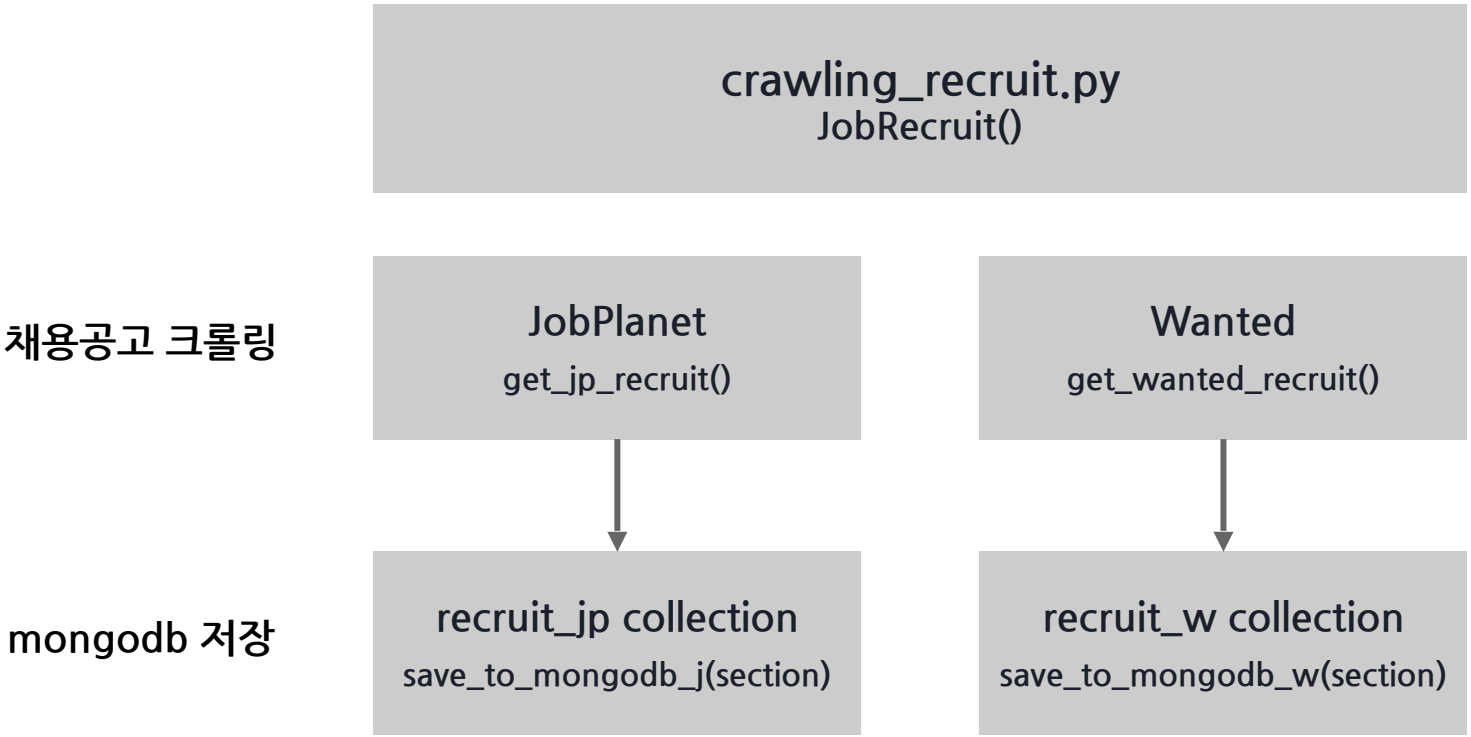
Procedure


Crontab을 이용해 자동화
채용공고 크롤링
JobPlanet, Wanted
기업리뷰 크롤링
JobPlanet
MongoDB 저장
공고/ 리뷰분석
Flask 구현
slack 메시지
전송


JOB OPENING CRAWLING
COMPANY REVIEW CRAWLING
OTHERS
OLD Jobplanet Recruit DB
Jobplanet Recruit Crawling
NEW Jobplanet Recruit DB
Set(Recruit-Review)
Company Name List (only.txt)
OLD Jobplanet Review DB
Review Crawling
NEW Jobplanet Review DB
Slack Message
OLD Wanted Recruit DB
Wanted Recruit Crawling
NEW Wanted Recruit DB
Set(OLD + NEW) - OLD
Flask
Make Web Pages
Analysis
AUTOMATION
aws
Crontab

Job Opening Crawling

JJ



Job Opening Crawling

get_jp_recruit()

1. session에 로그인정보 저장(ini 파일)
2. xpath로 링크 수집
3. BeautifulSoup으로 공고 내용 수집

get_wanted_recruit()

1. Selenium 으로 링크 수집
2. BeautifulSoup 으로 공고 내용 수집

save_to_mongodb_j()
save_to_mongodb_w()

1. get_jp_recruit()를 실행하여 mongodb에 저장

```
config = configparser.ConfigParser()
config.read('login.ini')
user = config.get(section, 'ID')
pwd = config.get(section, 'PW')
ip = config.get(section, 'IP')

account = f'mongodb://{user}:{pwd}@{ip}'
server = pymongo.MongoClient(account, 27017)
db = server.jobplanet

collection_c = db.recruit_jp
datas = self.get_jp_recruit()
ids = collection_c.insert(datas)
```

Job Opening Text Analysis

1. 불용어 사전 만들기 (ability_stopwords.txt)

2. NLTK를 이용하여 tokenize

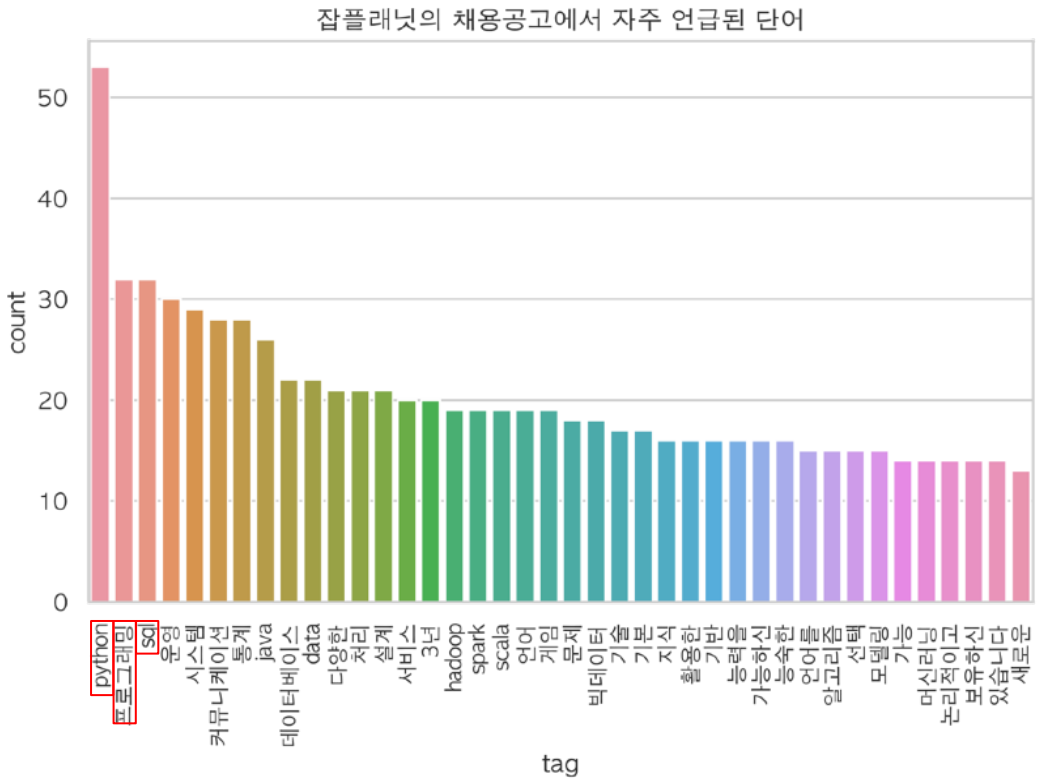
3. 빈도 파악

```
from nltk.tokenize import word_tokenize
from collections import Counter
from wordcloud import WordCloud

stop_words = "또는 관련 하신 대한 그에 준하는 사용 내용 높은 상기 최소 가진 자기 없으며 있는 있으신 혹은 빠르게 이상 분. 무관 경력
계신 함께 경험 경험자 갖는 소지 소지자 at of and with as such etc etc. us ms 있음 주요 in 이해 e.g 숙련 숙련자 능력 이상 이상의 무
관 경험이 활용 가능한 보유 필수 업무 거주자 우선 없는 없으신 취득 가지신 역량 등의 결격 보유자 이용한 사용한 이용 사용 분야 관련 대
한 능력 데이터 분석 개발 이해가 이해하고"

f = open('./ability_stopwords.txt', 'w')
f.write(stop_words)
f.close()

def ability_frequency(dataframe):
    ab_ls = list(dataframe.ability)
    ab_mass = ' '.join(ab_ls).replace(',', ' ').replace('•', ' ').replace(':', ' ').replace('&', ' ').replace('/', '
').replace('(', ' ').replace(')', ' ').replace('<', ' ').replace('>', ' ').replace('-', ' ').replace('*', '').lower()

    # 토큰화, 2글자 이상만 추출
    token = word_tokenize(ab_mass)
    token = [t for t in token if len(t) >= 2]

    # 불용어 제거
    stop = open('./ability_stopwords.txt', 'r').read().split(' ')
    token_rm = [t for t in token if not t in stop]

    # 사용빈도 카운트
    count = Counter(token_rm)
    tags = count.most_common(300)

    return tags
```

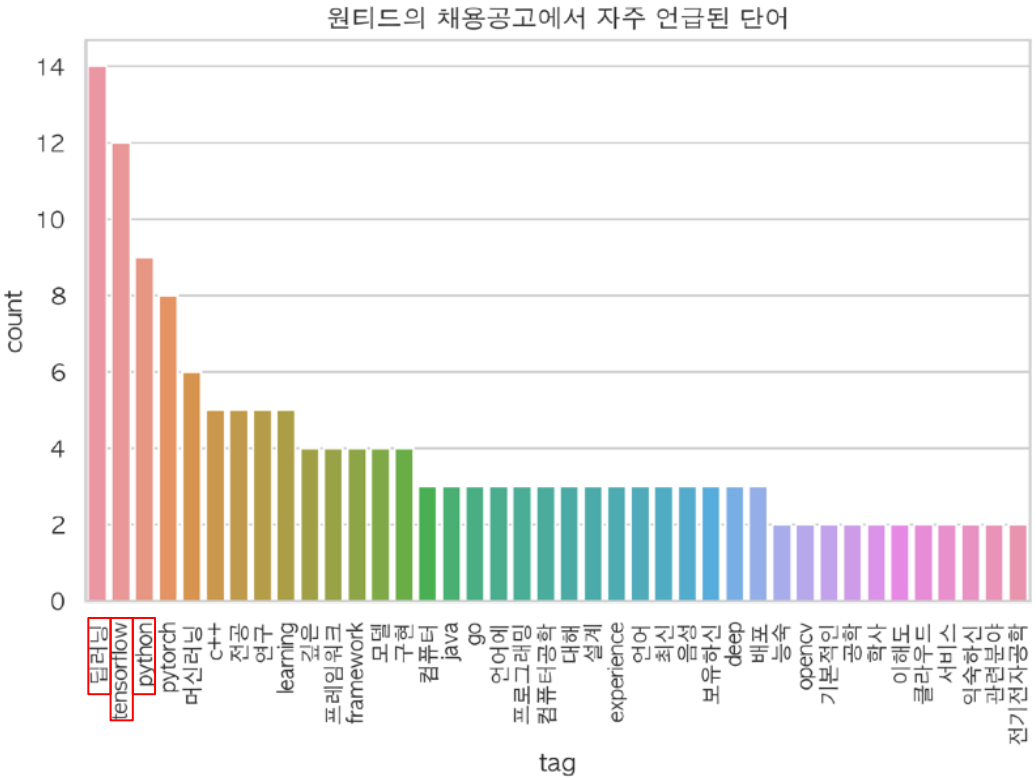
4. wordcloud로 시각화

```
def ability_visualizing(tags, platform):
    df = pd.DataFrame(tags)
    df.rename(columns={0: 'tag', 1: 'count'}, inplace=True)

    barplot = sns.barplot(x=df['tag'][:40], y=df['count'][:40], data=df[:40])
    plt.title(f'{platform}의 채용공고에서 자주 언급된 단어')
    plt.xticks(rotation=90, fontsize=10); plt.show()

    wordcolud = WordCloud(font_path='/System/Library/Fonts/Supplemental/AppleGothic.ttf', background_color='white',
colormap='Set2', width=600, height=600)

    cloud = wordcolud.generate_from_frequencies(dict(tags))
    cloudplot = plt.figure(figsize=(10, 8))
    # plt.title(f'{platform}의 채용공고에서 자주 언급된 단어')
    plt.axis('off'); plt.imshow(cloud); plt.show()

    return barplot, cloudplot
```

Job Opening Text Analysis [JopPlanet]



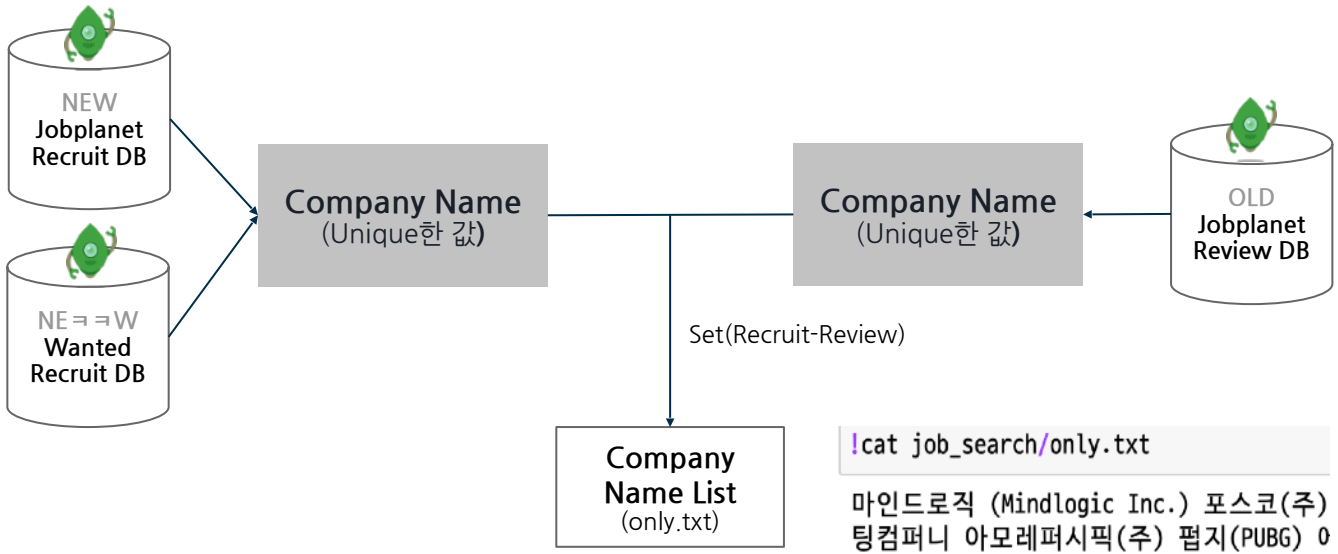
Job Opening Text Analysis [Wanted]



Company Review Crawling

JH

STEP 1. 리뷰 크롤링 해야하는 기업이름을 only.txt로 저장




```
!cat job_search/only.txt
```

```
마인드로직 (Mindlogic Inc.) 포스코(주)
팅컴퍼니 아모레퍼시픽(주) 펍지(PUBG) 0
(주)이스트소프트 큐픽스 넥슨네트웍스(주
(주) 와이더블유모바일(YWMobile)
```

Company Review Crawling

STEP 2. Bash for문을 사용하여, only.txt 에 저장된 기업 이름으로 리뷰 크롤링 실행

```
!cat run.sh
```

```
cd job_search
for i in `cat only.txt`
do
    echo "$i"
    scrapy crawl JobSearch -o job_planet_ds_review.csv -a company_name="$i"
done
```

```
!./run.sh
```

```
마인드로직
2020-04-30 17:33:59 [scrapy.utils.log] INFO: Scrapy 2.1.0 started (bot: job_search
2020-04-30 17:33:59 [scrapy.utils.log] INFO: Versions: lxml 4.5.0.0, libxml2 2.9.9
0.3.0, Python 3.7.6 (default, Jan  8 2020, 13:42:34) - [Clang 4.0.1 (tags/RELEASE_
19), cryptography 2.8, Platform Darwin-19.3.0-x86_64-i386-64bit
2020-04-30 17:33:59 [scrapy.utils.log] DEBUG: Using reactor: twisted.internet.sele
2020-04-30 17:33:59 [scrapy.crawler] INFO: Overridden settings:
{'BOT_NAME': 'job_search',
```

Company Review Crawling

STEP 3. 리뷰 크롤링을 위한 Scrapy 실행

1. 기업이름을 인자로, 리뷰 페이지 접근

2. While문을 이용해 전체 페이지를 크롤링

3. session을 이용해 로그인 정보(ini) 저장

4. 리뷰 페이지 링크 수집

5. Xpath를 이용해 리뷰 텍스트 수집

```
class Spider(scrapy.Spider):
    name="JobSearch"

    def __init__(self,company_name='구글코리아',**kwargs):
        self.company_name=company_name
        self.start_urls='https://www.jobplanet.co.kr/search?query={}&category=&_rs_con=welcome&_rs_act=index&_rs_element=main_search_bar'.forma
        super().__init__(**kwargs)

    def start_requests(self):
        url=self.start_urls
        yield scrapy.Request(url,callback=self.parse)

    def parse(self,response):
        item=JobSearchItem()
        link_first=response.xpath('//*[@id="mainContents"]/div[1]/div/div[2]/div[1]/div/a/@href').extract()[0]
        link_first=response.urljoin(link_first)
        item['link']=link_first

        #모든 page의 reveiw_text,person 크롤링
        page=1
        titles_ls,strength_ls,weakness_ls,wants_ls,person_ls=[],[],[],[],[]
        while True:
            link=link_first.split('info/')[0]+'reviews/'+self.company_name+'?page={}'.format(page)

            #link 접속
            login_url = 'https://www.jobplanet.co.kr/users/sign_in'
            login_data = {'user': {'email': config.get('section1', 'ID'), 'password': config.get('section1', 'PW'), 'remember_me':'true'}}
            session = requests.session()
            req = session.post(login_url, json = login_data)
            req = session.get(link)
            response_s=TextResponse(req.url, body=req.text, encoding='utf-8')

            #review_text
            elements=response_s.xpath('//*[@id="viewReviewsList"]/div/div/div/section/div/div[2]/div/div[1]/h2/text()').extract()
            titles = [element.replace("\n", "").strip() for element in elements]
            titles = " ".join(titles)
            #title의 길이가 0일때까지 page의 수를 증가시킨다.
            if len(titles)==0: break
            titles_ls.append(titles)

            elements=response_s.xpath('//*[@id="viewReviewsList"]/div/div/div/section/div/div[2]/div/dl/dd[1]/span/text()').extract()
            strengths = [element.replace("\n", "").replace("\r","").strip() for element in elements]
            strengths = " ¦ ".join(strengths)
            strength_ls.append(strengths)
```

Company Review Text Analysis

1. 불용어 사전 만들기 (review_stopwords.txt)

2. DB의 리뷰 텍스트 불러오기

3. NLTK를 이용하여 tokenize, 빈도 파악

4. wordcloud로 시각화, 이미지 저장(.png)

```
    def bring_reviews(self):
        server = pymongo.MongoClient(self.account, 27017)
        db = server.jobplanet
        review_a = db.collection
        collection = db.review
        rv = pd.DataFrame(list(collection.find()))
        rv = rv[['company_name', 'review_num', 'stats', 'strength', 'title', 'want', 'weakness', 'person']]
        rv.review_num = rv.review_num.apply(lambda x: int(x))
        rv = rv[rv.review_num > 0]
        rv['company_name2'] = rv.company_name.apply(lambda x:  ''.join(re.findall('[가-힣]+', x.replace('(주)', '').replace('(유)', '').replace(' ',
'').replace('(', '').replace(')', ''))))

        return rv

    def pros_and_cons(self, company):
        """
        return company pros and cons wordcloud images
        input =>
            - rv : review DataFrame
            - company : company_name2
        """
        rv = bring_reviews('ju')
        pros = list(rv[rv.company_name2 == company].strength)[0][0]
        pros = re.sub('[0-9][.]', '', pros)
        pros = pros.replace('|', ' ').replace('/', ' ').replace('(', ' ').replace(')', ' ').replace('-', ' ')

        cons = list(rv[rv.company_name2 == company].weakness)[0][0]
        cons = re.sub('[0-9][.]', '', cons)
        cons = cons.replace('|', ' ').replace('/', ' ').replace('(', ' ').replace(')', ' ').replace('-', ' ')

        # 토큰화, 2글자 이상만 추출
        token_p = word_tokenize(pros)
        token_p = [t for t in token_p if len(t) >= 2]
        token_c = word_tokenize(cons)
        token_c = [t for t in token_c if len(t) >= 2]

        # 불용어 제거
        stop = open('./review_stopwords.txt', 'r').read().split(' ')
        token_pr = [t for t in token_p if not t in stop]
        token_cr = [t for t in token_c if not t in stop]

        # 사용빈도 카운트
        count_p = Counter(token_pr)
```

Company Review Text Analysis [Pros & Cons]

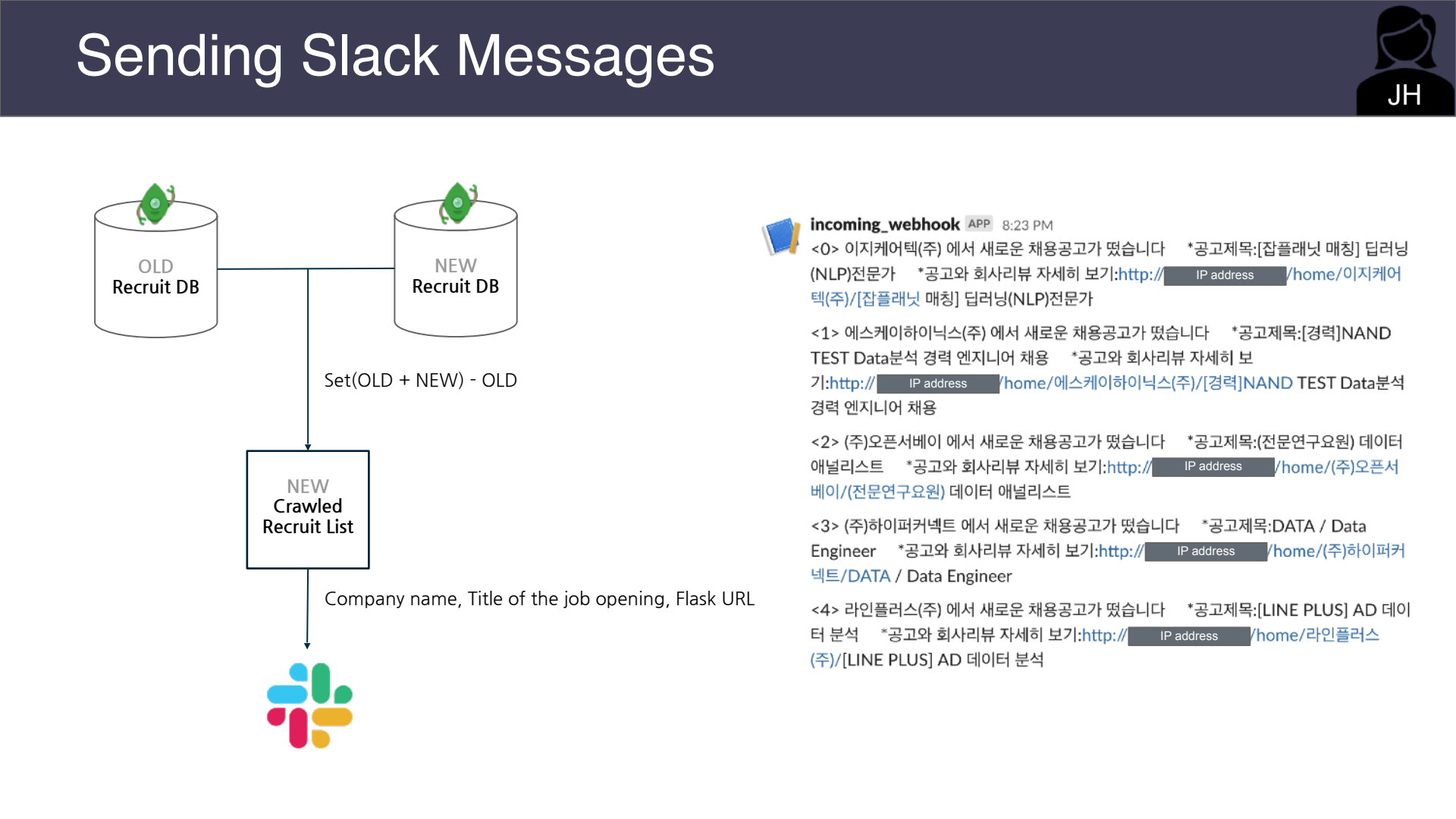
JJ

[시각화 예시] 네이버.png, 아모레퍼시픽.png

〈네이버의 장단점〉

〈아모레퍼시픽의 장단점〉

Sending Slack Messages



incoming_webhook APP 8:23 PM

<0> 이지케어텍(주) 에서 새로운 채용공고가 떴습니다 *공고제목:[잡플래닛 매칭] 딥러닝(NLP)전문가 *공고와 회사리뷰 자세히 보기:http://IP address/home/이지케어텍(주)/[잡플래닛 매칭] 딥러닝(NLP)전문가

<1> 에스케이하이닉스(주) 에서 새로운 채용공고가 떴습니다 *공고제목:[경력]NAND TEST Data분석 경력 엔지니어 채용 *공고와 회사리뷰 자세히 보기:http://IP address/home/에스케이하이닉스(주)/[경력]NAND TEST Data분석 경력 엔지니어 채용

<2> (주)오픈서베이 에서 새로운 채용공고가 떴습니다 *공고제목:(전문연구요원) 데이터 애널리스트 *공고와 회사리뷰 자세히 보기:http://IP address/home/(주)오픈서베이/(전문연구요원) 데이터 애널리스트

<3> (주)하이퍼커넥트 에서 새로운 채용공고가 떴습니다 *공고제목:DATA / Data Engineer *공고와 회사리뷰 자세히 보기:http://IP address/home/(주)하이퍼커넥트/DATA / Data Engineer

<4> 라인플러스(주) 에서 새로운 채용공고가 떴습니다 *공고제목:[LINE PLUS] AD 데이터 분석 *공고와 회사리뷰 자세히 보기:http://IP address/home/라인플러스(주)/[LINE PLUS] AD 데이터 분석

Sending Slack Messages

1. 크롤링 진행 전, OLD Recruit DB를 pickle객체로 저장

```
class jobplanet_all():

    def get_old_recruit_df(self):
        #크롤링 전 mongodb에 저장된 데이터를 데이터프레임으로 저장
        #old_mongodb 를 df로 만들기(pandas가 다루기 편하기 때문!)
        old_recruit_df_jp = pd.DataFrame(list(collection_c_jp.find()))
        old_recruit_df_w = pd.DataFrame(list(collection_c_w.find()))
        #pickle로 저장
        old_recruit_df_jp.to_pickle("job_search/old_recruit_df_jp.pkl")
        old_recruit_df_w.to_pickle("job_search/old_recruit_df_w.pkl")
```

2. 새롭게 추가된 채용공고를 반환 [Set(OLD + NEW) - OLD]

```
#new_minus_old : df의 차집합 구하기
def new_minus_old(self,old,new):
    #old와 new를 행으로 합치기
    old_new_concat=pd.concat([old,new],keys=list(new.columns),axis=1)
    #old의 multiindex만 뽑아내기
    list(old_new_concat.columns[:int(len(old_new_concat.columns)/2)])
    #old의 모든 행이 NAN인 행 뽑아내기
    null_df=old_new_concat[list(old_new_concat.columns[:int(len(old_new_concat.columns)/2)])].isnull()
    #null_df의 Multiindex 없애기
    null_df.columns = null_df.columns.droplevel()
    #new에서 null_df의 값을 반환
    #new_minus_old : 우리가 원하는 값! new에만 있는 값!
    new_minus_old=new[null_df].dropna(how='all').drop_duplicates()
    return new_minus_old

def new_mongo_db(self,which_old_recruit_df):
    old_recruit_df = pd.read_pickle("job_search/{}.pkl".format(which_old_recruit_df))

    if which_old_recruit_df=='old_recruit_df_jp':
        #old+new된 newmongodb
        new_recruit_df = pd.DataFrame(list(collection_c_jp.find()))

    elif which_old_recruit_df=='old_recruit_df_w':
        #old+new된 newmongodb
        new_recruit_df = pd.DataFrame(list(collection_c_w.find()))

    new_recruit_df=new_recruit_df.drop('_id',axis=1)
    #new_mongodb_df의 duplicate를 drop 하여 다시 저장
    new_recruit_df=new_recruit_df.drop_duplicates()
    new_recruit_df=new_recruit_df.reset_index(drop=True)
    #result_mongodb_df= new_mongodb_df-old_mongodb_df
    result_recruit_df=self.new_minus_old(old_recruit_df,new_recruit_df)
    result_recruit_df=result_recruit_df.reset_index(drop=True)
    return result_recruit_df
```

3. (2)의 함수를 JopPlanet, Wanted 각각 수행

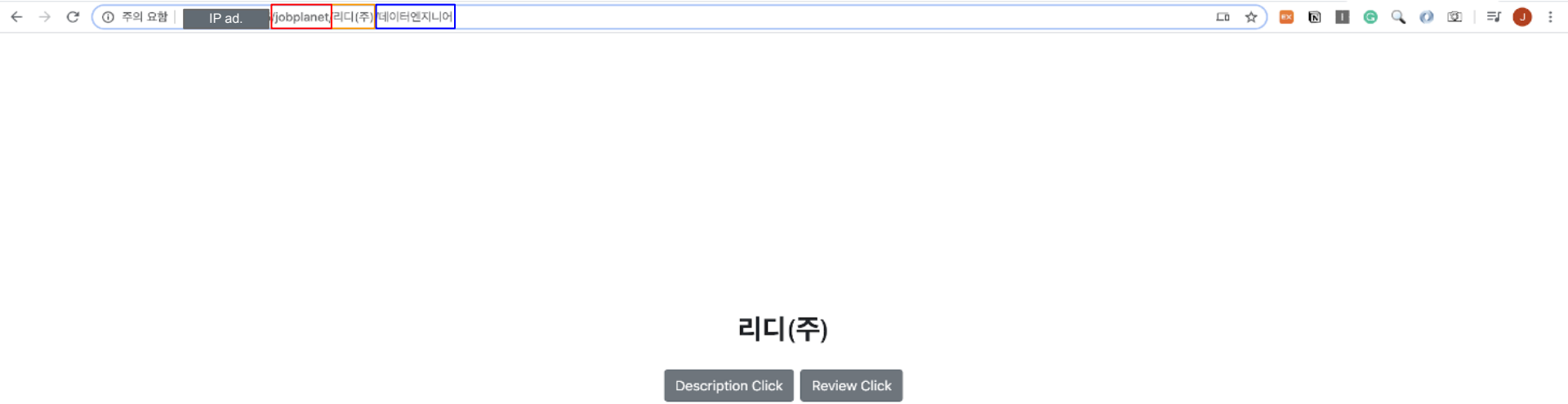
Make Web Pages with Flask

1. DB에 저장된 Recruit, Review 데이터를 불러옴

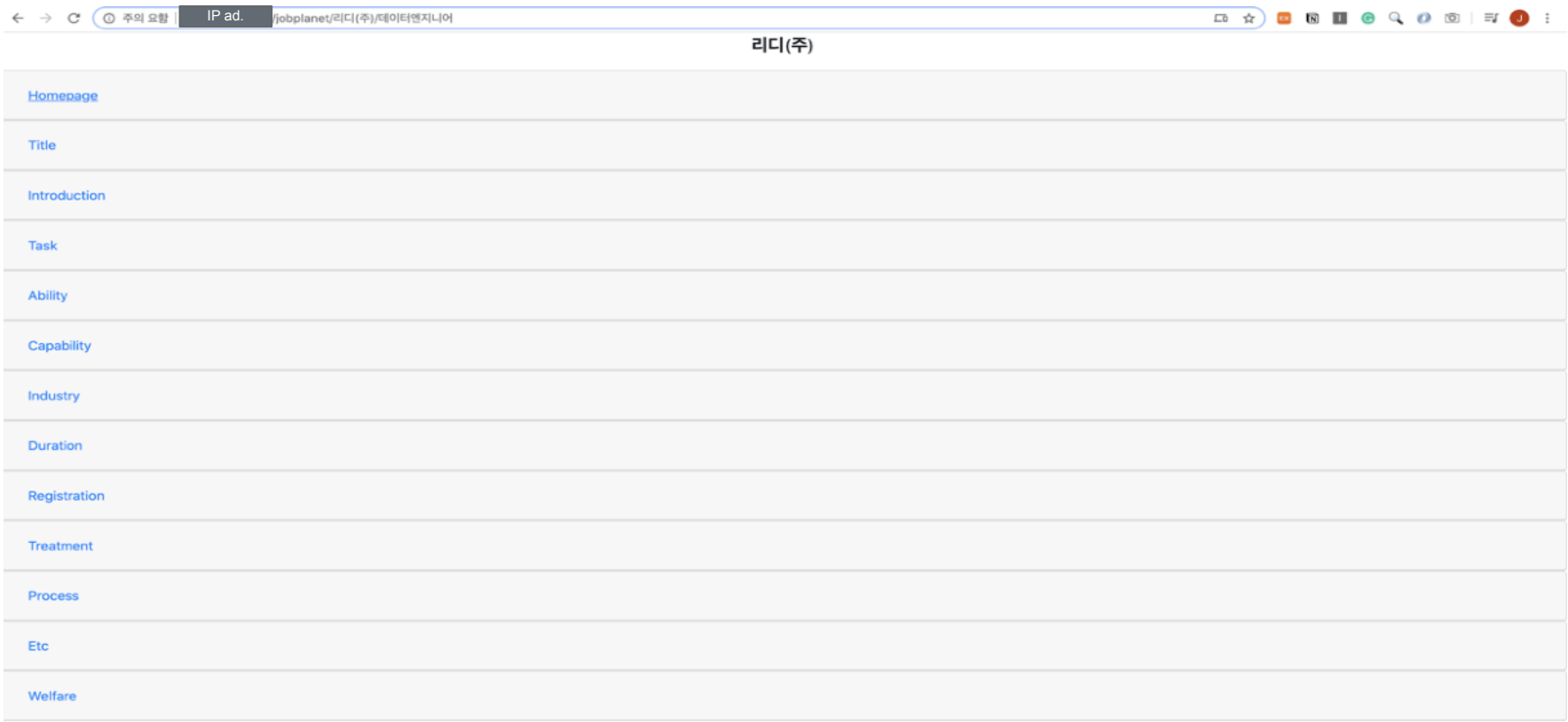
2. HTML과 연동(자세한 html 코드는 Git에 있음!)

```
%%writefile jobflask/jobflask.py
import pickle
import os
import pymongo
from flask import *
import configparser
from mongodb_ini import mongodb

app=Flask(__name__)


@app.route("/raw_review/<company_name>/<title2>")
def raw_review(company_name,title2):
    #jobplanet db의 review 테이블 접근
    db=mongodb()
    collection_re=db.review
    result= list(collection_re.find({'company_name':'{}'.format(company_name)}))[0]
    del result['_id']
    return jsonify(result)

#jobplanet raw_recruit_data
@app.route("/raw_recruit_jp/<company_name>/<title2>")
def raw_recruit_jp(company_name,title2):
    #jobplanet db의 recruit_jp 테이블 접근
    db=mongodb()
    collection_c_jp=db.recruit_jp
    result=list(collection_c_jp.find({"$and":[{"title2":'{}'.format(title2)},{"company_name":"{}".format(company_name)}]}))[0]
    del result['_id']
    return jsonify(result)

#wanted raw_recruit_data
@app.route("/raw_recruit_w/<company_name>/<title2>")
def raw_recruit_w(company_name,title2):
    #jobplanet db의 recruit_w테이블 접근
    db=mongodb()
    collection_c_w=db.recruit_w
    result=list(collection_c_w.find({"$and":[{"title2":'{}'.format(title2)},{"company_name":"{}".format(company_name)}]}))[0]
    del result['_id']
    return jsonify(result)

@app.route("/jobplanet/<company_name>/<title2>")
def jobplanet(company_name,title2):
    return render_template('index_jp.html',company_name=company_name,title2=title2)

@app.route("/wanted/<company_name>/<title2>")
def wanted(company_name,title2):
    return render_template('index_w.html',company_name=company_name,title2=title2)

app.run(debug=True)
```

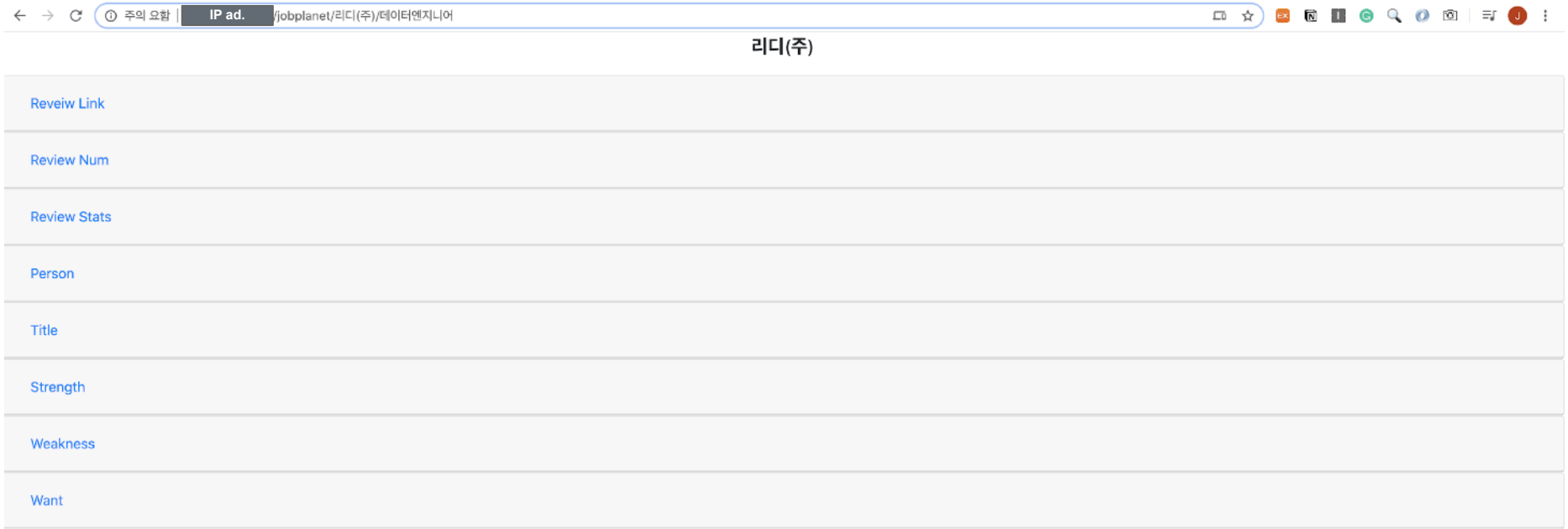
An Example of Web Pages [Main - JobPlanet]



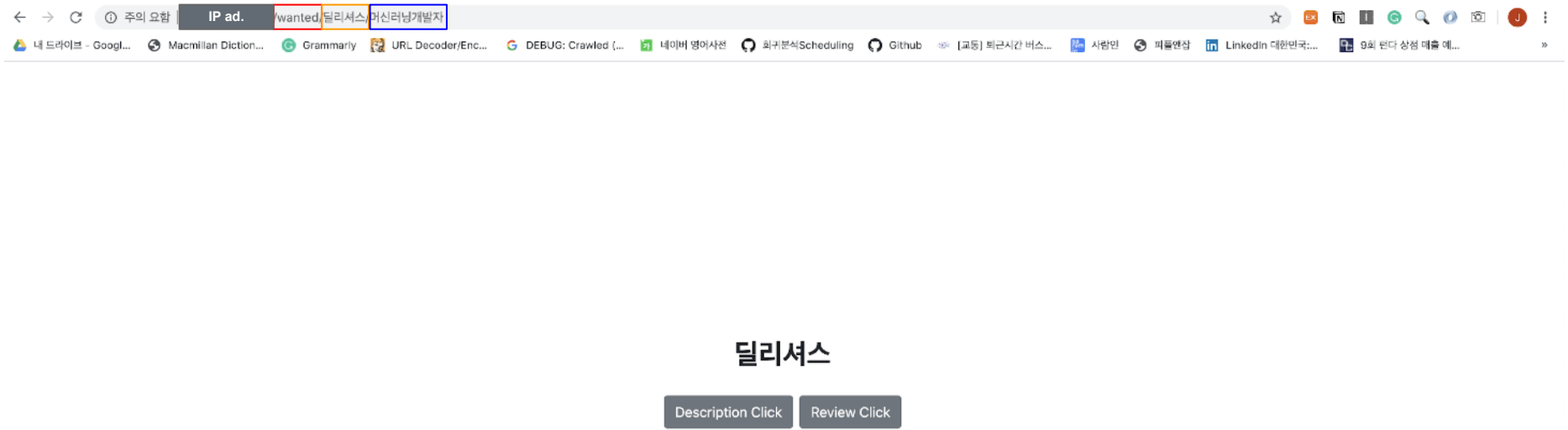
An Example of Web Pages [Job Opening – JobPlanet]



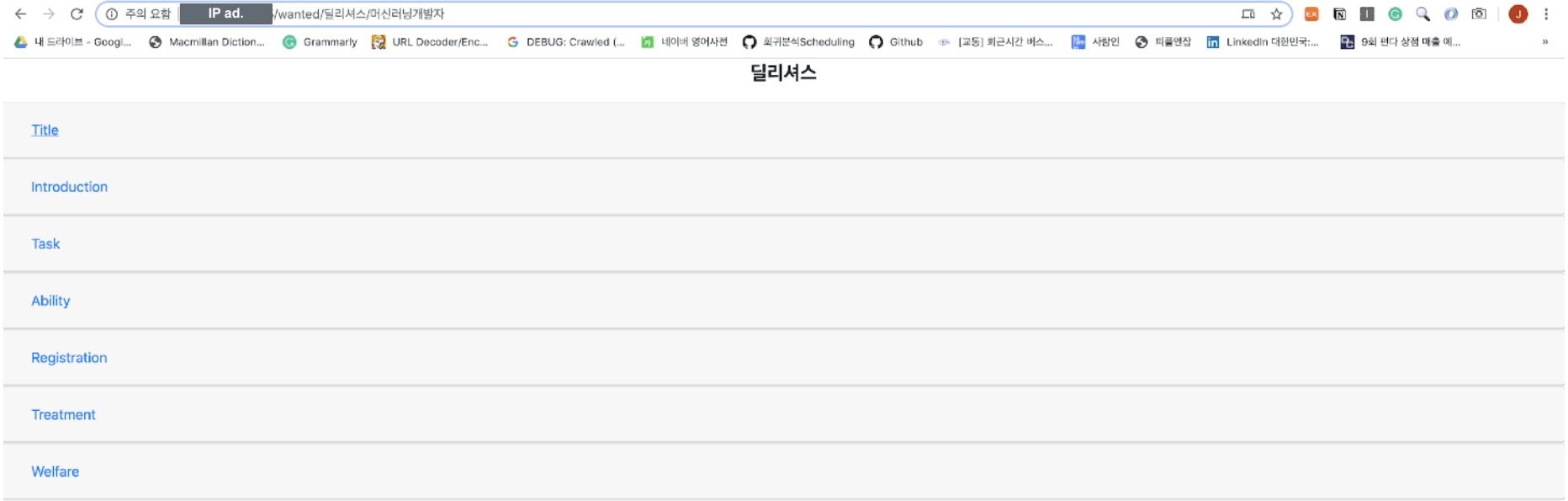
An Example of Web Pages [Company Review - JobPlanet]



An Example of Web Pages [Main - Wanted]



An Example of Web Pages [Job Opening – Wanted]



Automating the Process Using Crontab

매일 오전 10시 10분부터
20분 간격으로 실행

```
01 10 * * * /home/ubuntu/.pyenv/versions/python3/bin/python /home/ubuntu/python3
/notebook/06_scrapy/crawling_project/to_review_crawling.py
21 10 * * * /home/ubuntu/python3/notebook/06_scrapy/crawling_project/run.sh
41 10 * * * /home/ubuntu/.pyenv/versions/python3/bin/python /home/ubuntu/python3
/notebook/06_scrapy/crawling_project/to_slack.py
```

cat to_review_crawling.py

```
# -*- coding: utf-8 -*-
from jobplanet_all import *
jobplanet=jobplanet_all()
jobplanet.to_review_crawling()
```

cat run.sh

```
cd job_search
for i in `cat only.txt`
do
    echo "$i"
    scrapy crawl JobSearch -o job_planet_ds_review.csv -a company_name="$i"
done
```

cat to_slack.py

```
from jobplanet_all import *
jobplanet=jobplanet_all()
jobplanet.to_slack()
print("slack 메세지 보내기 완료")
```

Jobplanet_all.py

```
def to_review_crawling(self):
    self.get_old_recruit_df() #크롤링 이전, 몽고db의 테이블을 객체로 저장
    self.recruit_crawling() #공고 크롤링
    self.review_crawling() #리뷰 크롤링

def to_slack(self):
    #jobplanet
    result_recruit_df_jp=self.new_mongo_db('old_recruit_df_jp') #크롤링 이전의 몽고db 객체와 크롤링 이후의 몽고db 데이터프레임을 비교함
    self.slack(result_recruit_df_jp,'Jobplanet') #크롤링 이후의 몽고 db에 새롭게 추가된 내용을 slack 메시지로 전송
    #wanted
    result_recruit_df_w=self.new_mongo_db('old_recruit_df_w')
    self.slack(result_recruit_df_w,'Wanted')
```

Retrospective

- What Could Be Improved

- 분석 결과를 DB에 저장하여 웹 페이지로 구현
- 데이터가 쌓이면, 기업의 채용 주기를 파악하여 이직률을 유추