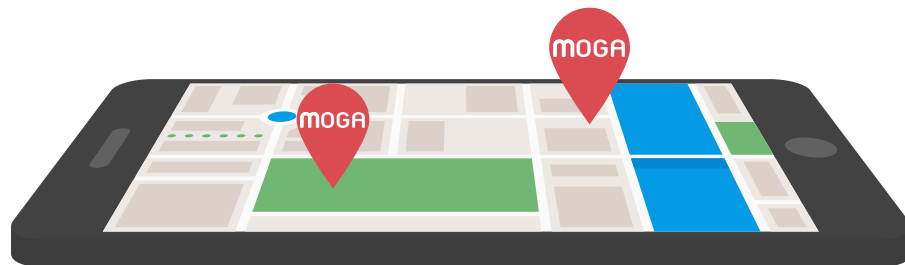


MOGA

Every price you need



맛집추천 챗봇 모두의 가격



빅데이터 분석을 위한 자바 개발자 과정

Overview



데이터 수집 및 전처리

망고플레이트 맛집 정보 및 리뷰 크롤링
데이터베이스 저장



카카오톡 챗봇 만들기

Django, Amazon web service, github
카카오톡 플러스친구



안드로이드 맛집 추천 앱

Android Studio 앱 만들기
Mecab 품사분석 및 감성분석 사전 구축

SW tool



01. 데이터 수집 : 웹크롤링

1. 파이썬을 이용해 맛집 리뷰 사이트 망고플레이트에서 가게 정보 크롤링

```
def priceinfo(a):
    search=driver.find_element_by_id("main-search")
    search.send_keys(a)
    xpath_search="//html/body/main/article/header/fieldset/input"
    driver.find_element_by_xpath(xpath_search).click()
    l = 0
    for n in tqdm_notebook(range(0, 10)):
        driver.find_element_by_xpath(xpath_page[n]).click()
        for i in range(0, 10):
            b = i + 1

            time.sleep(0.3)

            driver.find_element_by_xpath(xpath_restaurant[i]).click()

            time.sleep(0.3)

            html=driver.page_source
            soup=BeautifulSoup(html, 'html.parser')

            try:
                time.sleep(0.3)
                for n in range(len(soup.find_all("li", "Restaurant_Menulitem"))):
                    restaurant_name.append(soup.find("li", "restaurant_name").get_text())
                    restaurant_menu.append(soup.find_all("span", "Restaurant_Menu")[n].get_text())
                    restaurant_price.append(soup.find_all("span", "Restaurant_MenuPrice")[n].get_text().replace(",",""))
                    reference_index.append(b)
            except AttributeError:
                for n in range(len(soup.find_all("li", "Restaurant_Menulitem"))):
                    restaurant_name.append(soup.find("li", "restaurant_name").get_text())
                    restaurant_menu.append("-")
                    restaurant_price.append("-")

            time.sleep(0.3)

            data = {'상호명':restaurant_name, '메뉴':restaurant_menu, '가격':restaurant_price, 'reference index':reference_index}
            df = pd.DataFrame(data)

            driver.execute_script("window.history.go(-1)")
            time.sleep(0.3)
            l = l + 10

        df.to_csv("priceinfo" + a + ".csv", mode='w', encoding='euc-kr')
        print(df)

priceinfo("강남 맛집")
```

: Webdriver와
selenium 으로
웹 크롤러 만들기

정형데이터

MANGO
PLATE



곰다비워드 4.7
논현동 - 베이커리
₩ 120,874 / 108



정돈 프리미엄 4.7
가로수길 - 카스 요리
₩ 40,997 / 77



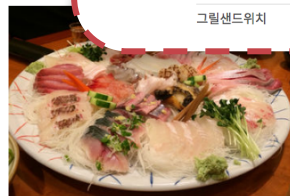
이치에 4.7
신사/압구정 - 이자카야 / 오명 / 꼬치
₩ 268,072 / 107



다운타운 4.7
신사/압구정 - 프랑스 음식
₩ 50,941 / 48



파씨오네 4.7
신사/압구정 - 프랑스 음식
₩ 59,642 / 48



김수사 4.7
논현동 - 회 / 스시
₩ 44,545 / 41

01. 데이터 수집 : 웹크롤링



2. 검색기준 선정

: 서울특별시 자치구 + 행정동 + 맛집 핫플레이스 별로 수집

3. Csv 파일로 저장

종로구
청운효자동
사직동
삼청동
부암동
평창동
무악동
교남동
가회동
종로1.2.3.4가동
종로5.6가동
이화동
혜화동
창신제1동

서울행정구역.txt

25개 자치구와 424개 행정동

```
In [1]: search_list = ['강남 맛집', '송파 맛집', '강동 맛집', '서초 맛집', '관악구 맛집', '동작구 맛집',  
                        '금천구 맛집', '영등포구 맛집', '구로구 맛집', '양천구 맛집', '강서구 맛집',  
                        '광진구 맛집', '성동구 맛집', '용산구 맛집', '마포구 맛집', '중구 맛집', '종로 맛집',  
                        '마포구 맛집', '서대문구 맛집', '중랑구 맛집', '노원구 맛집', '강북구 맛집', '은평구 맛집',  
                        '성북구 맛집', '도봉구 맛집']  
  
search_list_more = ['예술의전당 맛집', '교대 맛집', '남부터미널 맛집', '샤로수길 맛집', '광화문 맛집',  
                    '홍대 맛집', '연남동 맛집', '연희동 맛집', '잠실 맛집', '석촌호수 맛집', '서울대입구 맛집',  
                    '합정 맛집', '망원동 맛집', '성수 맛집', '이태원 맛집', '남산 맛집', '명동 맛집',  
                    '건대 맛집', '제주도 맛집', '노량진 맛집', '한강진 맛집', '신촌 맛집', '성대여대 맛집']
```



01. 데이터 수집 : 데이터베이스 저장



4. csv 파일을 Mysql 에 저장 (Placeinfo priceinfo 각각)



Mysql 정규화 (1NF)

한 가게 당 여러 메뉴와 가격 정보
불필요한 중복 제거

	A	B	C	D	E	F	G
1	id	상호명	주소	전화번호	음식종류	가격대	평점
2		0 정돈 프리미엄	서울시 강남구 신사동 519-4	02-549-0924	까스 요리	만원~2만원	4.8
3		1 베이커치즈타르트	서울시 서초구 반포동 19-3	02-3479-1204	카페 / 디저트	만원 미만	4.8
4		2 리틀앤머치	서울시 강남구 삼성동 10-8	02-545-1023	카페 / 디저트	만원 미만	4.7
5		3 꿈다비쥬드	서울시 강남구 논현동 257	02-511-8456	베이커리	만원 미만	4.7
6		4 다운타운너	서울시 강남구 신사동 666-25	070-8820-3696	브런치 / 버거 / 샌드위치	만원 미만	4.7

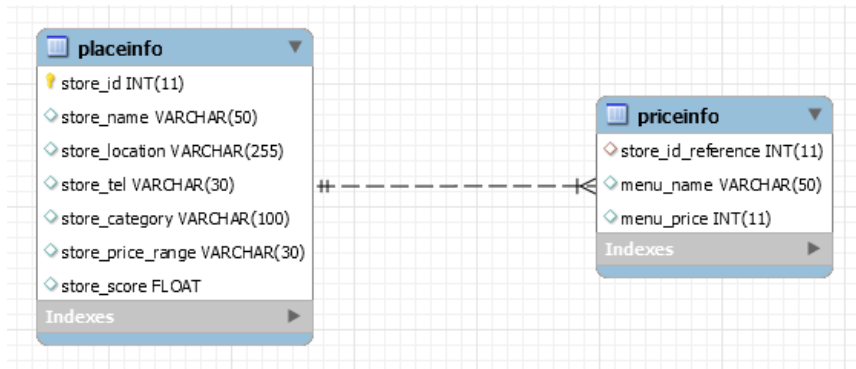
▼ placeinfo.csv

	A	B	C
1	store_id reference	가격	메뉴
2		1	2800 치즈타르트
3		2	8500 바이올렛 쿼츠
4		2	9000 화이트 초콜릿 돔
5		2	7500 피스타치오 트로피칼
6		2	7500 아이리쉬 크림
7		2	8500 런던 포그
8		3	3000 아메리카노
9		3	5500 에그샐러드 샌드위치
10		3	7000 그릴샌드위치
11		4	7800 베이컨치즈버거
12		4	9300 아보카도버거
13		4	6800 치즈버거
14		4	6300 갈릭버터 프라이즈
15		4	7300 제네럴 쏘 치킨

▶ priceinfo.csv

```
create table placeinfo(
  store_id int
  ,store_name varchar(50)
  ,store_location varchar(255)
  ,store_tel varchar(30)
  ,store_category varchar(100)
  ,store_price_range varchar(30)
  ,store_score float
  ,primary key(store_id));

create table priceinfo(
  store_id_reference int
  ,menu_name varchar(50)
  ,menu_price int
  ,foreign key(store_id_reference)
  references placeinfo(store_id));
```



비정형데이터



-

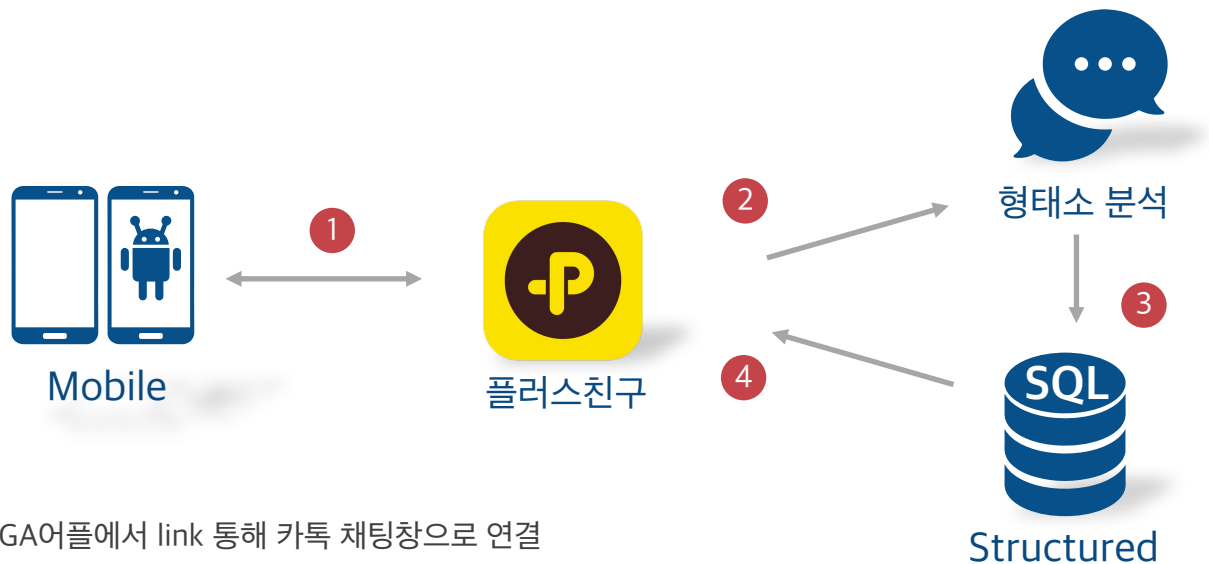
- 7

2. 챗봇 만들기



02. 챗봇 만들기: MOGA bot overview

MOGA



- 1 MOGA어플에서 link 통해 카톡 채팅창으로 연결
- 2 파이썬에서 mecab을 통해 사용자 언어 분석
- 3 Mysql 데이터베이스에서 필요한 정보 가져오기
- 4 사용자에게 정보를 제공

02. 챗봇 만들기: chatbot 구축 툴

웹프레임워크 : Django
서버 : AWS(아마존 웹 서비스)



순서

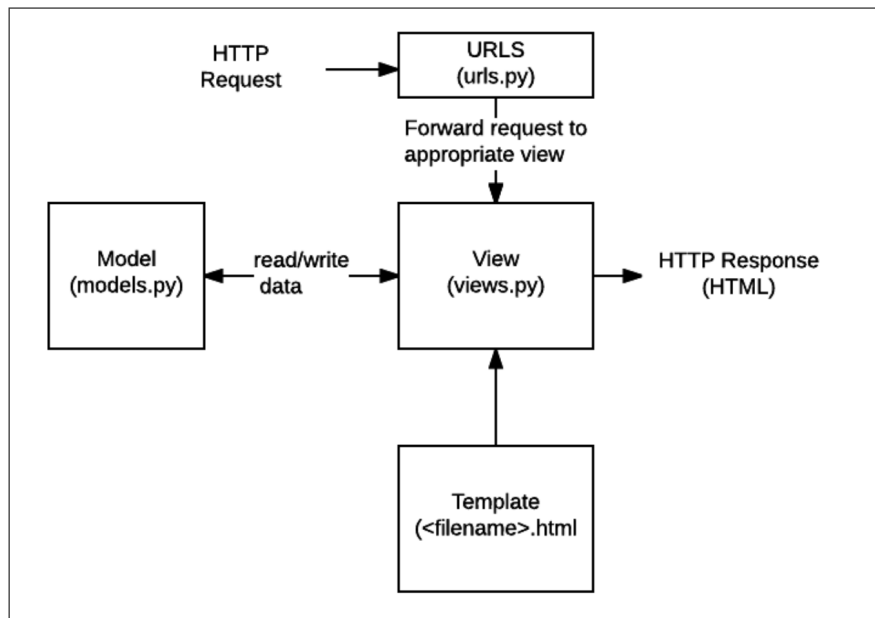
1. 파이썬과 Django를 이용해서 간단한 채팅봇 만들기
2. Github 사용하기
3. 아마존 웹서비스에서 django 배포
4. 카카오톡 플러스친구를 통해 챗봇 구현하기

02. 챗봇 만들기

1 Django와 python으로 챗봇 만들기

Django

: 보안이 우수하고 유지보수가 편리한 웹사이트를 신속하게 개발하는 하도록 도움을 주는 파이썬 웹 프레임워크



02. 챗봇 만들기

MOGA

1 Django와 python으로 챗봇 만들기

1. project-settings.py
데이터베이스 설정

2. project-urls.py url 설정

3. app-urls.py url 설정

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'moga',  
        'HOST': 'localhost',  
        'USER': 'root',  
        'PASSWORD': 'd!lghehd36',  
        'PORT': '3306',  
    }  
}
```



```
from django.contrib import admin  
from django.conf.urls import url, include
```

```
urlpatterns = [  
    url(r'^moga/', include('moga.urls')),  
    url(r'^admin/', admin.site.urls),  
]
```

```
from django.conf.urls import url
```

```
from . import views
```

```
urlpatterns = [  
    url(r'^keyboard/', views.keyboard),  
    url(r'^message', views.message)
```

02. 챗봇 만들기

1 Django와 python으로 챗봇 만들기

4. 받은 메시지를 parsing

```
@csrf_exempt
def message(request):
    m=Mecab()

    json_str=((request.body).decode('utf-8'))
    received_json_data=json.loads(json_str)
    datacontent_str=received_json_data['content']

    datacontent_split=re.split(' ', datacontent_str)
    datacontent=m.morphs(datacontent_split)
```

5. parsing한 메시지를 바탕으로 알맞은 함수 호출

```
elif '메뉴' in datacontent_split:
    return priceinfo(datacontent_split[0])
```

```
def priceinfo(name):
    sql = "select pr.menu_name, pr.menu_price from placeinfo as pl join priceinfo as pr on pl.store_id = pr.store_id_reference where pl.store_name = '" + name + "'"
    db = pymysql.connect(host='localhost', user='root', password='d1ghehd36', db='moga', charset='utf8')
    cursor = db.cursor()
    cursor.execute(sql)
    row = cursor.fetchall()

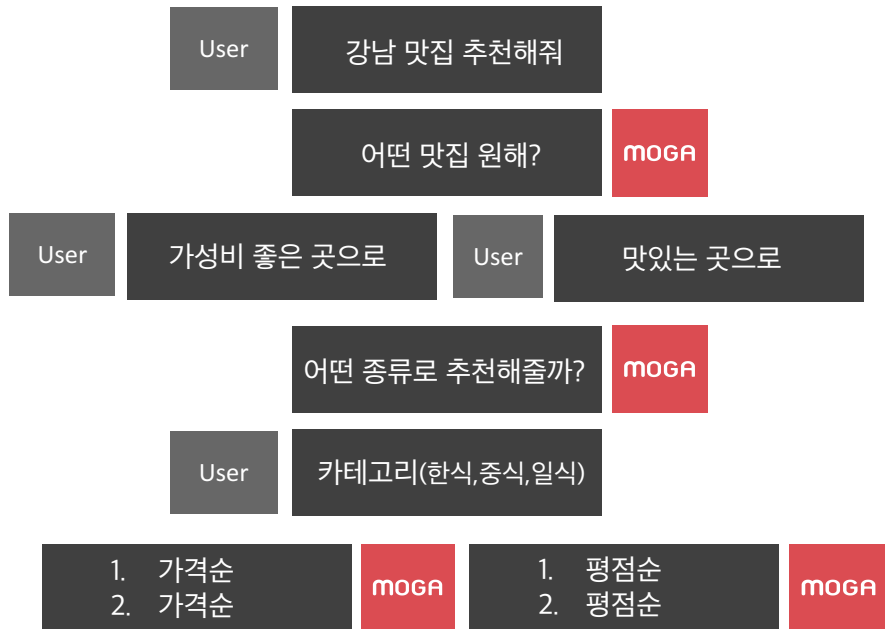
    a=[]
    for n in range(len(row)):
        a.append(row[n][0]+' : ' + str(row[n][1]) + '원')

    return JsonResponse({
        'message': {
            'text': str(a).replace(", ", "\n")
        },
        'keyboard': {
            'type': 'text'
        }
    })
```

6. 함수로 처리한 후 사용자에게 보여준다

02. 챗봇 만들기: chatbot flow

1 Django와 python으로 챗봇 만들기



02. 챗봇 만들기: 파이썬 상세코드

1 Django와 python으로 챗봇 만들기

Views.py(1)

```
def keyboard(request):  
    return JsonResponse({  
        'type': 'text'  
    })
```

Step1. Request가 들어오면 json형식으로 변환

Views.py(2)

```
@csrf_exempt  
def message(request):  
    m=Mecab()  
  
    json_str=((request.body).decode('utf-8'))  
    received_json_data=json.loads(json_str)  
    datacontent_str=received_json_data['content']  
  
    datacontent_split=re.split(' ',datacontent_str)  
    datacontent=m.morphs(datacontent_str)  
    text_data.append(datacontent_str)  
    length = len(text_data)
```

Views.py(3)

```
def placeinfo(name):  
    sql="select store_name from placeinfo"  
  
    db=pymysql.connect(host='localhost', user='root', password='q1ghhd98', db='moga', charset='utf8')  
    cursor=db.cursor()  
    cursor.execute(sql)  
    row=cursor.fetchall()  
  
    for n in range(len(row)):  
        if name in row[n][0]:  
            sql="select store_name, store_category, store_tel, store_url from placeinfo where store_name = " + name  
            cursor.execute(sql)  
            row=cursor.fetchall()  
  
            return JsonResponse({  
                'message': {  
                    'text': 이름 + " : " + str(row[0][0]) + " 종류 : " + str(row[0][1]) + " 전화번호 : " + str(row[0][2])  
                    + " 지도 열기 : " + str(row[0][3])  
                },  
                'keyboard': {  
                    'type': 'text'  
                }  
            })  
  
    else:  
        return JsonResponse({  
            'message': {  
                'text': 입력된 맛집 정보가 없습니다. 다시 입력해 주세요!  
            },  
            'keyboard': {  
                'type': 'text'  
            }  
        })
```

Step2. Message 구현 부분
Json으로 받은 데이터를 파이썬 언어로 반환

Step3. 필요한 함수를 정의하고
(가격순, 평점순으로 sorting)

Views.py(4)

```
elif '맛있는곳' in datacontent_split:  
    return scoreinfo(datacontent_split[0])  
  
elif '평점순으로' in datacontent_split:  
    return scoreinfo(datacontent_split[0])  
  
elif datacontent_str in category_list:  
    return categoryinfo(datacontent_str)  
  
else:  
    return placeinfo(datacontent_str)
```

Step4. 상황에 따라 적절한 함수를 호출하
면 적절한 답변이 Json형식으로 return됨

02. 챗봇 만들기

MOGA

2 Github 사용하기

vygudrbs / projectmoga12

Code Issues Pull requests Projects Wiki Insights Settings

No description, website, or topics provided.

Add topics

7 commits 1 branch 0 releases

Branch: master New pull request Create new file Upload file

vygudrbs Add files via upload

.idea	moga
Gangnamdata	Add files via upload
moga12	moga
projectmoga12	moga
static/admin	moga
venv	moga
infoallGangnam.csv	Add files via upload
manage.py	moga

Branch: master projectmoga12 / projectmoga12 /

pyo moga

..

__pycache__ moga

__init__.py moga

settings.py moga

urls.py moga

wsgi.py moga

23 lines (20 sloc) | 815 Bytes

```
1 """projectmoga12 URL Configuration
2
3 The 'urlpatterns' list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/2.0/topics/http/urls/
5 Examples:
6 Function views
7     1. Add an import: from my_app import views
8     2. Add a URL to urlpatterns: path('', views.home, name='home')
9 Class-based views
10    1. Add an import: from other_app.views import Home
11    2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13    1. Import the include() function: from django.urls import include, path
14    2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.conf.urls import url, include
18
19 urlpatterns = [
20     url(r'^moga12/', include('moga12.urls')),
21     url(r'^admin/', admin.site.urls),
22 ]
```

02. 챗봇 만들기

2 Github 사용하기

Branch: master ▾ projectmoga12 / moga12 /

Create new file Upload files Find file History

pyo moga Latest commit 048586b 4 days ago

..

__pycache__	moga	4 days ago
migrations	moga	4 days ago
__init__.py	moga	4 days ago
admin.py	moga	4 days ago
apps.py	moga	4 days ago
models.py	moga	4 days ago
tests.py	moga	4 days ago
urls.py	moga	4 days ago
views.py	moga	4 days ago

Branch: master ▾ projectmoga12 / moga12 / urls.py

Find file Copy path

pyo moga 048586b 4 days ago

0 contributors

8 lines (6 sloc) | 147 Bytes

Raw Blame History

```
1 from django.conf.urls import url
2
3 from . import views
4
5 urlpatterns=[
6     url(r'^keyboard/', views.keyboard),
7     url(r'^message', views.message),
8 ]
```

MOGA



on github

YourGitHubName/Thematic



on your computer

PathToYourRepo/Thematic

Github Clone

: 원격에서는 소스를 수정할 수 없으므로
Fork한 저장소를 작업할 로컬 머신에 받는 과정

Create new file

Upload files

Find file

Clone or download ▾

Clone with HTTPS ?

Use SSH

Use Git or checkout with SVN using the web URL.

https://github.com/vygudrbs/projectmc



Open in Desktop

Download ZIP

02. 챗봇 만들기

3 Amazon Web Service에서 django 배포

아마존 웹서비스 등록 -> 아마존 EC2에서 코드 배포

☐	Name ▾	인스턴스 ID ▲	인스턴스 유형 ▾	가용 영역 ▾	인스턴스 상태 ▾	상태 검사 ▾	경보 상태	퍼블릭 DNS(IPv4)
☒		i-0d8599c25422f0530	t2.micro	ap-northeast-2c	● running	✔ 2/2 검사 통과	없음	 ec2-52-79-111-167
☐		i-0f72d1c335ecf151b	t2.micro	ap-northeast-2c	● stopped		없음	

인스턴스: **i-0d8599c25422f0530** 퍼블릭 DNS: **ec2-52-79-111-167.ap-northeast-2.compute.amazonaws.com**

02. 챗봇 만들기

3 Amazon Web Service에서 django 배포

ubuntu@ip-172-31-17-179: ~

```
ubuntu@ip-172-31-17-179:~$ dir
Gangnamdata  mecab.sh  projectmoga
ubuntu@ip-172-31-17-179:~$
```

▶ github를 통해 코드를 올린 후
아마존 웹서비스로 코드를 복사하면
projectmoga가 생김

ubuntu@ip-172-31-17-179: ~/projectmoga

```
ubuntu@ip-172-31-17-179:~/projectmoga$ python3 manage.py runserver 0.0.0.0:8000
Performing system checks...

System check identified no issues (0 silenced).
June 19, 2018 - 00:16:47
Django version 2.0.6, using settings 'projectmoga.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

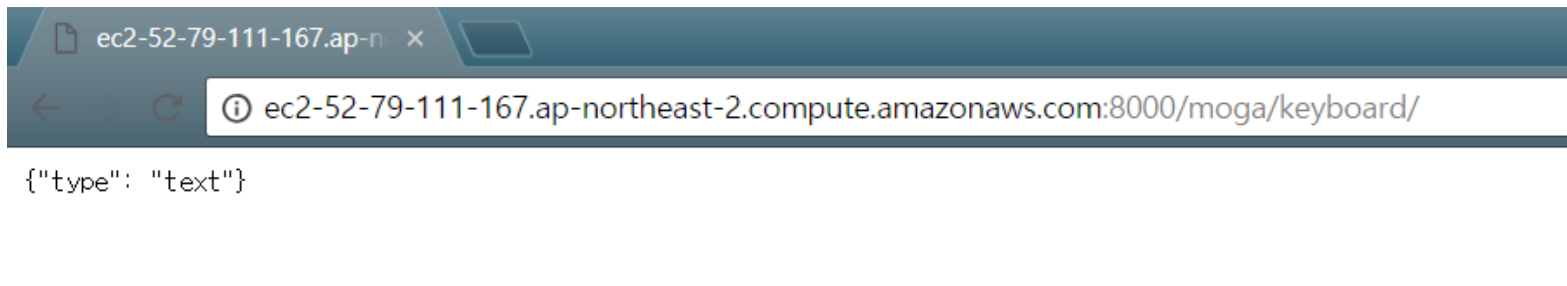
아마존 웹서비스를 통해
Django배포

02. 챗봇 만들기

3 Amazon Web Service에서 django 배포

아마존 웹서비스의 퍼블릭DNS:8000/app/(keyboard)로 접속

Success!



Json 형식으로 알맞게 장고 코드가 작성됨

02. 챗봇 만들기

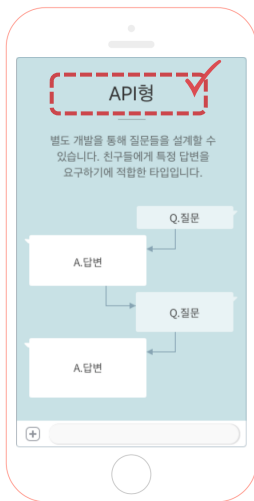
4 Kakao Talk 플러스친구를 통해 챗봇 구현하기

: 카카오톡 자동응답 API를 통해 사용자가 입력한 input 값에 대해 json으로 답변
: Keyboard와 message 구현

 플러스친구 관리자센터



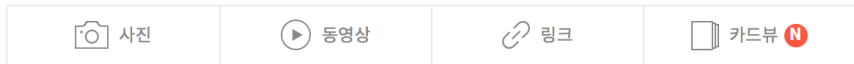
시작하기



종지하기



모두의가격의 새 이야기를 들려주세요.



[카드뷰란?](#)

02. 챗봇 만들기

4 Kakao Talk 플러스친구를 통해 챗봇 구현하기

API형

별도의 개발의 통해 특정 답변을 요구하는 형태의 질문들을 설계하는 타입입니다.

앱 등록

앱 이름

moga

앱 URL

<http://ec2-52-79-111-167.ap-northeast-2.compute.amazonaws.com:8000/m>

API 테스트

Required*

keyboard OK
{ "type": "text" }

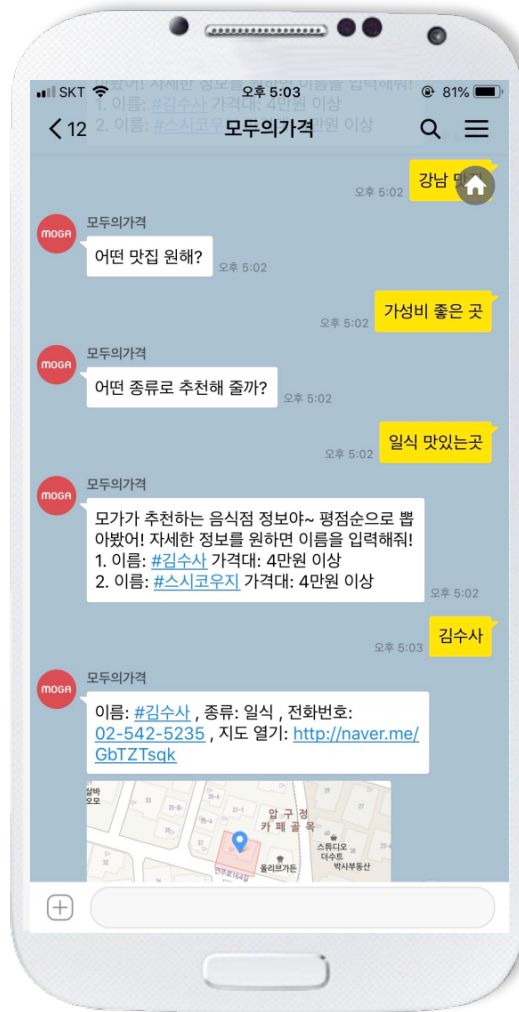
앱 설명

간단한 챗봇입니다.

1. 앱 URL에 배포한 django URL를 입력
2. API테스트를 거쳐 오류가 없다면 서비스를 이용 준비

```
ubuntu@ip-172-31-17-179: ~/projectmoga
[19/Jun/2018 00:32:34] "GET /moga/keyboard/ HTTP/1.1" 200 16
[19/Jun/2018 00:32:36] "POST /moga/message HTTP/1.1" 200 74
[19/Jun/2018 00:32:44] "POST /moga/message HTTP/1.1" 200 94
[19/Jun/2018 00:33:07] "POST /moga/message HTTP/1.1" 200 119
[19/Jun/2018 00:33:16] "POST /moga/message HTTP/1.1" 200 182
[19/Jun/2018 00:33:20] "POST /moga/message HTTP/1.1" 200 333
```

카카오톡을 통해 받은 각각의 메시지가
GET, POST 형식으로 출력됨



Talk to
MOGA Bot !



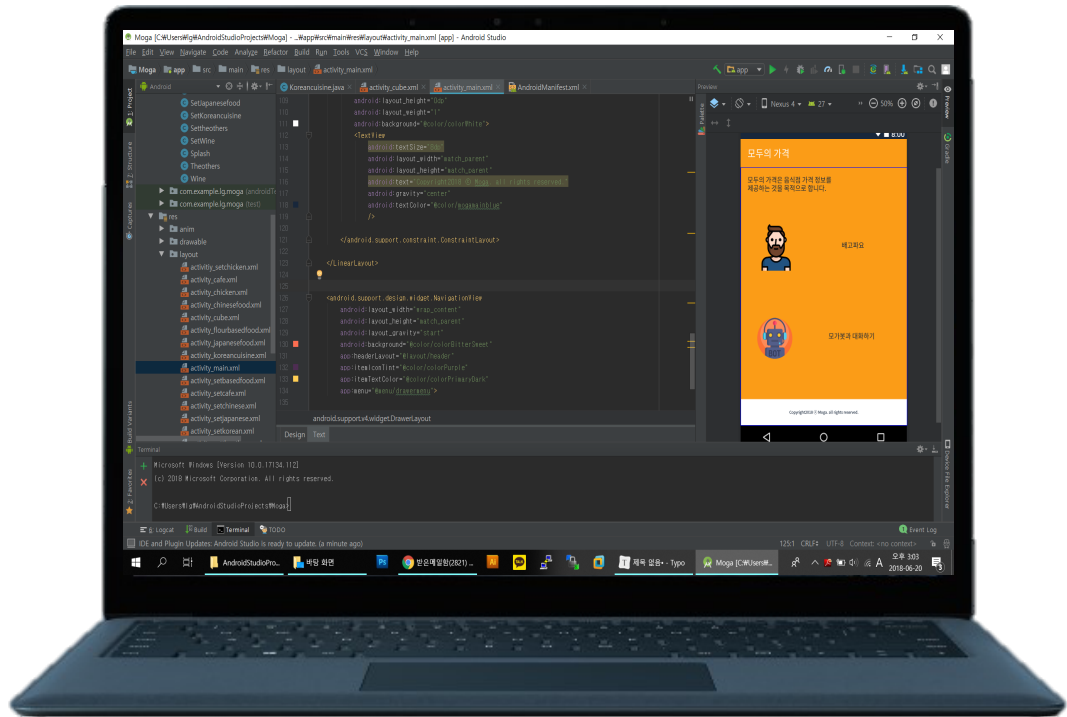
3. Android App

제작하게 된 배경은 모가봇 접속을 위한 사용자 인터페이스가 필요하다고
생각했습니다. 앱을 통해서 카카오톡 플러스친구로 연결 되도록 구성하였고
향후 모두의가격 확장성을 위해서 앱을 제작하게 되었습니다.

03. 안드로이드 앱 : Android Studio

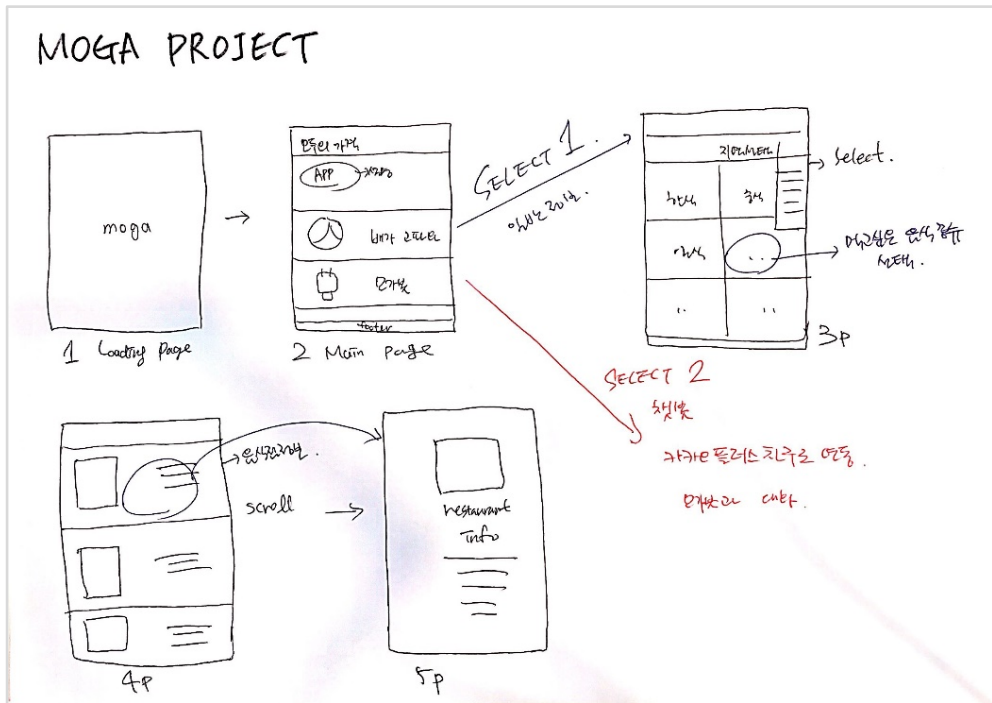
MOGA

사용 프로그램



03. 안드로이드 앱 :UI draft

MOGA



◀ 초안



앱 구상(UI/UX)

두 가지 방법으로 원하는 정보를 얻을 수 있음
(<배고파요> 클릭 or <모가봇과 대화하기> 클릭)
Kakao 커넥트

03. 안드로이드 앱 : Design color & Logo

MOGA



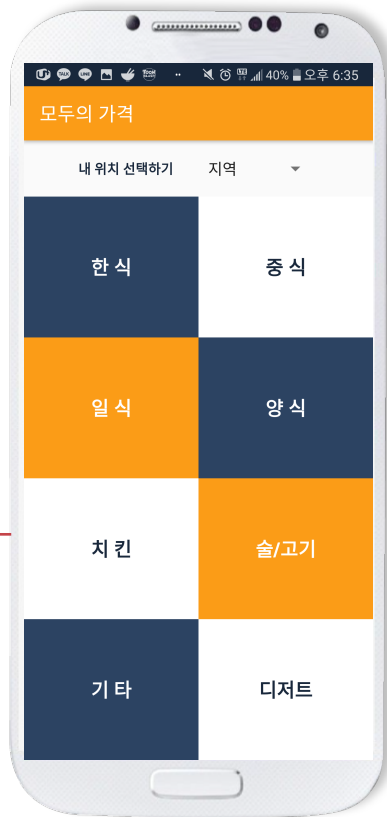
App Logo



Color

오렌지, 남색, 흰색

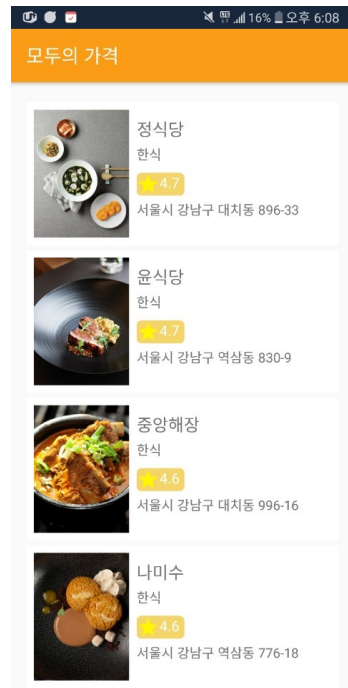
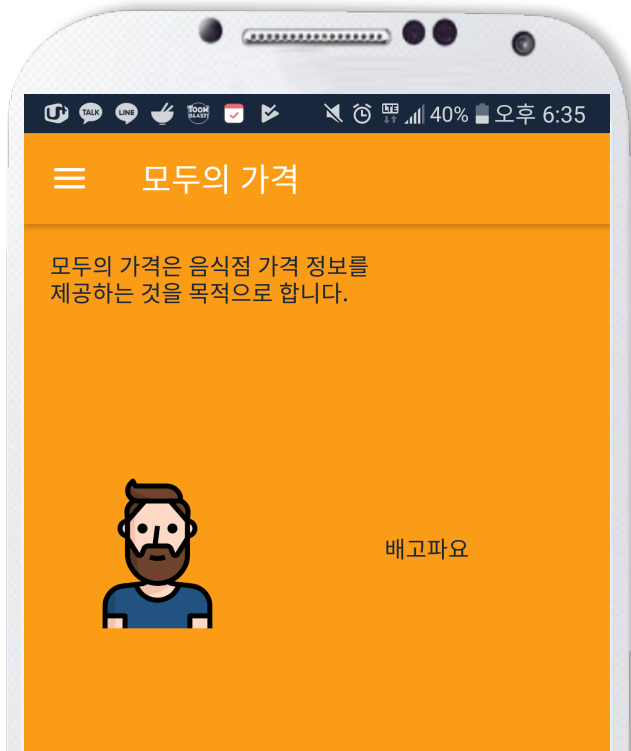
친숙한 디자인으로
별도의 가이드 없이 직관적으로 사용 가능



03. 안드로이드 앱 : UI/UX디자인

When?

1. 내 주변 음식점 정보를 한 눈에 보고 고르고 싶을 때
배고파요 클릭 -> 자신의 지역 -> 음식 종류 선택

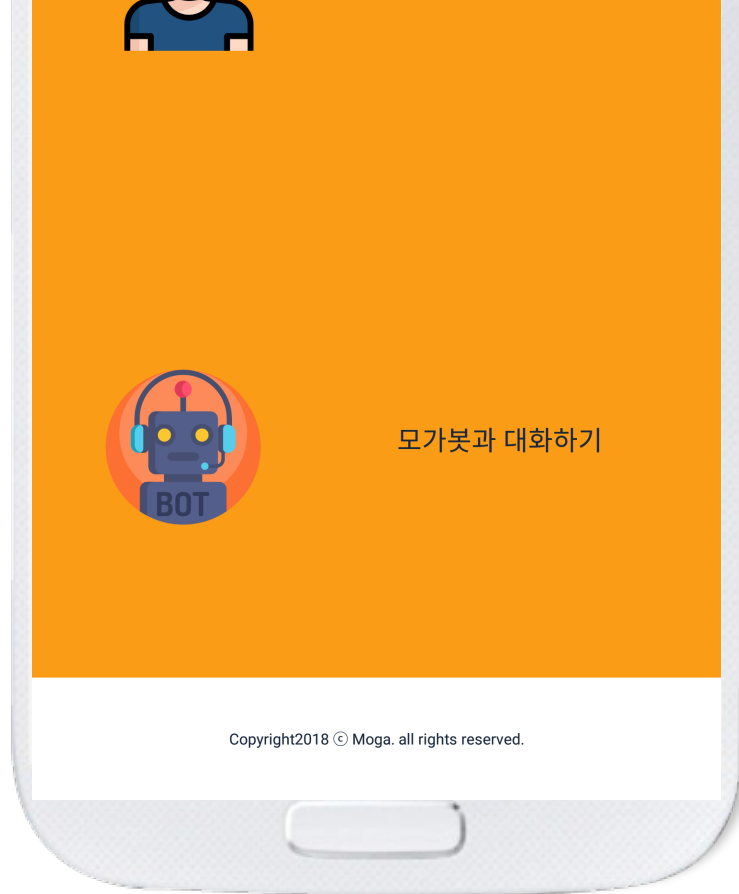


03. 안드로이드 앱 : UI/UX디자인

MOGA

2. 뭘 먹을 지 모르겠을 땐 모가봇에게 물어보기

모가 카카오톡 채팅방으로 이동
가격순, 평점순으로 추천

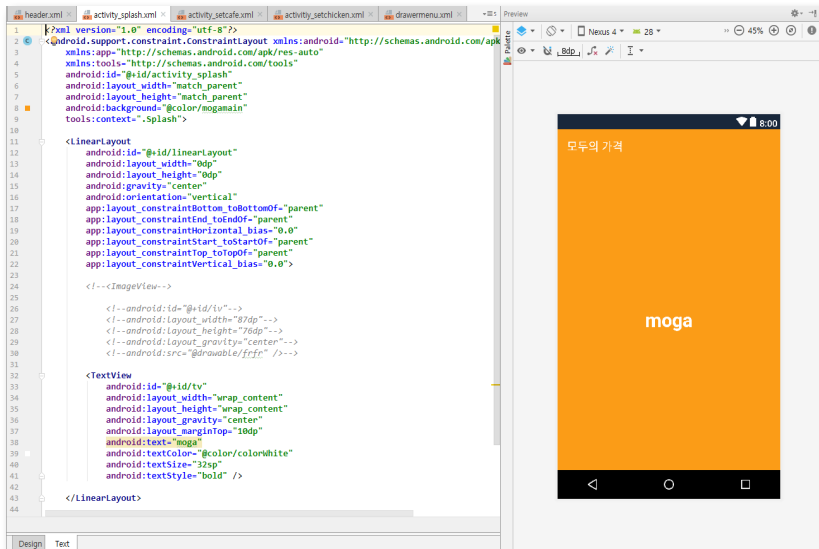


03. 안드로이드 앱 : 제작과정

MOGA

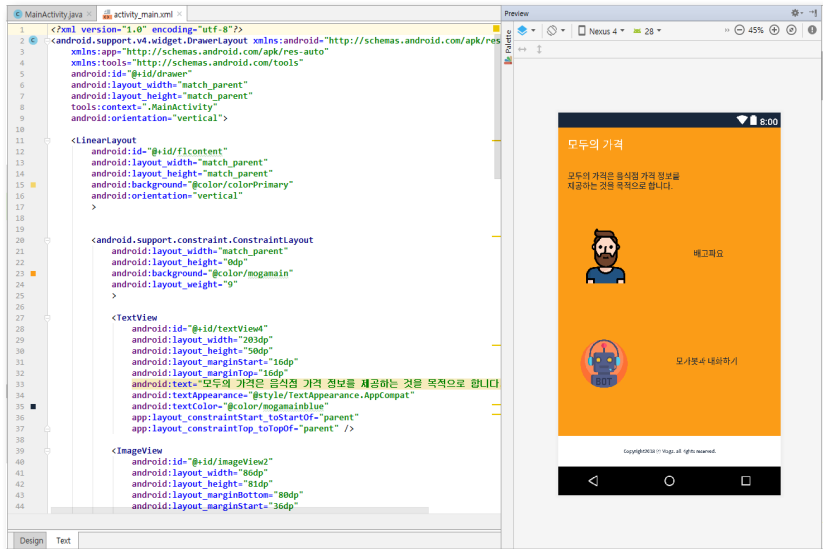


로딩/ 메인 페이지



Loading page

로딩(splash)화면



Main page

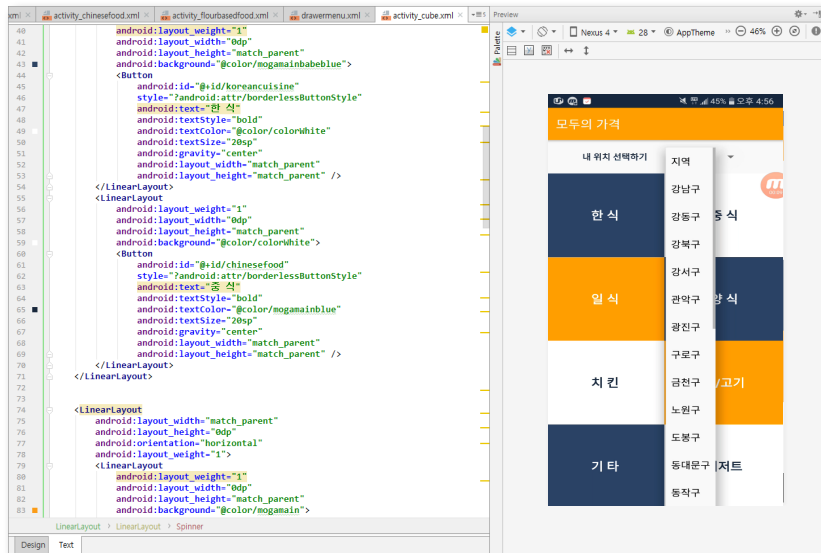
다음 선택지로 갈 수 있는 button과
Kakao를 이용해 채팅할 수 있는 버튼 추가

03. 안드로이드 앱 : 제작과정

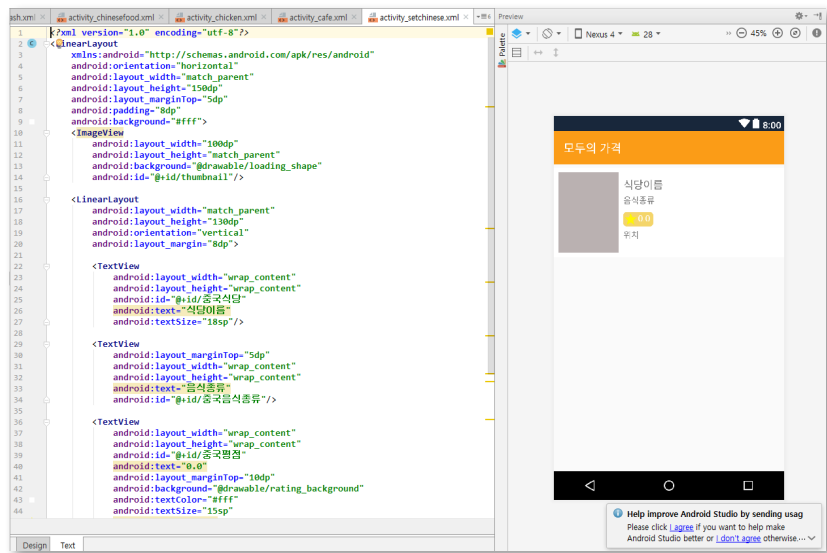
MOGA



Spinner layout/Recycler view



Spinner layout
지역 선택란 추가



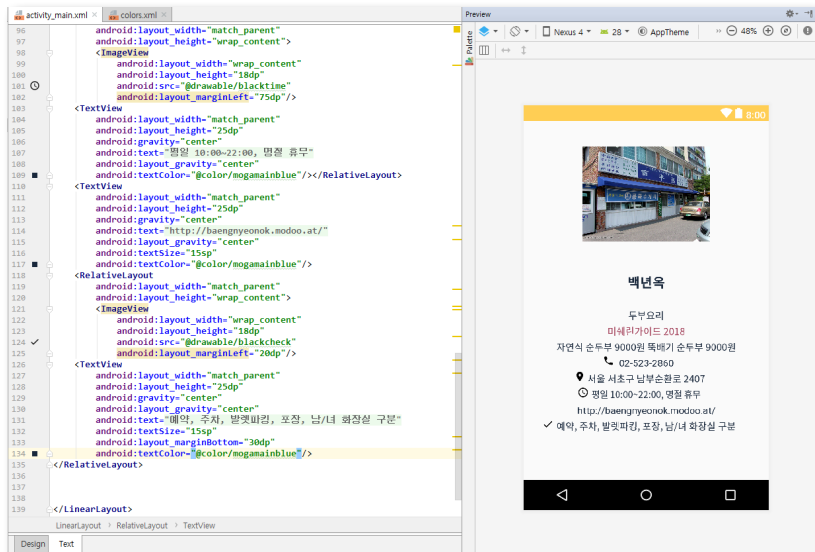
RecyclerView
주변 음식점 리스트를 한눈에

03. 안드로이드 앱 : 제작과정

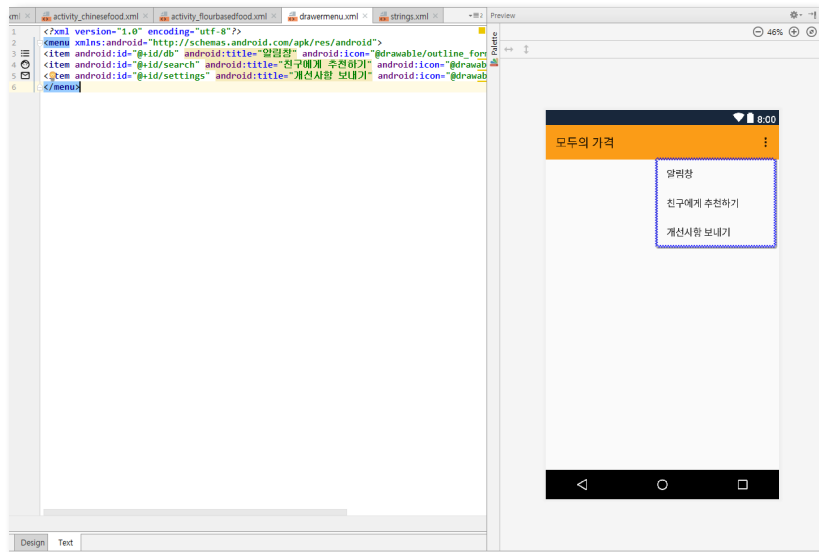
MOGA



Subpage/etc



Subpage
음식점 세부 정보



Navigation drawer
알림창, 친구에게 추천하기, 개선사항 보내기 기능
Beta버전에서는 비활성화

03. 안드로이드 앱 : 제작과정

MOGA



크롤링한 데이터 앱에 올리기

앱에서 보여줄 CSV 파일을 JSON으로 convert

```
id,restaurant,menu,location,tel,main_menu,price,average_price
0.청춘 프티미엄,계곡 요리,서울시 강남구 신사동 519-4,02-549-0924,책방,18,000원,2만원
1.베이커리드미스트,카페 / 디저트,서울시 서초구 반포동 19-3,02-3479-1204,디저트,7,800원,만원 미만
2.김홍겸마치,카페 / 디저트,서울시 강남구 삼성동 10-8,02-545-1023,베이커리,믹스,8,500원,만원 미만
3.클라비티드,베이커리,서울시 강남구 논현동 257,02-511-9456,아메리카노,3,000원,만원 미만
4.다올루,보통커피 / 베이 / 샌드위치,서울시 강남구 신사동 666-25,070-8820-3696,아보카도버거,9,300원,만원 미만
5.파에요디,프랑스 음식,서울시 강남구 신사동 646-23,02-546-7719,한지 코스,41,800원,4만원 이상
6.간수시,회 / 스키,서울시 강남구 논현동 4-19,02-542-5235,한지 사치미,50,000원,4만원 이상
7.강북이끼,이끼카페 / 호텔 / 호텔,서울시 강남구 신사동 651-5,02-540-9669,스치오,19,000원,만원~2만원
8.농민백담,쌀 / 찰 / 찰,서울시 강남구 대치동 896-33,02-555-9603,국밥,7,000원,만원 미만
9.농민백담,쌀 / 찰 / 찰,서울시 강남구 대치동 896-33,02-555-9603,국밥,7,000원,만원 미만
10.로얄,빵 / 디저트,서울시 강남구 삼성동 83-02,6377-2014,인제,73,000원,만원~4만원
11.노이베리,베이커리,서울시 강남구 신사동 612,02-546-7588,레몬링,5,000원,만원 미만
12.이끼고,카페 / 디저트,서울시 강남구 대치동 909-15,02-553-0416,꽃가게,5,500원,만원 미만
13.임구원,구급,카페 / 디저트,서울시 강남구 신사동 567-21,02-516-3643,죽,5,500원,만원 미만
14.계곡,고기 요리,서울시 강남구 논현동 107-11,02-515-6080,삼겹살,10,000원,만원~2만원
15.통영,고기 요리,서울시 강남구 논현동 97-7,02-3448-5646,통영육회,17,000원,3만원~4만원
16.가네이로,프티미엄,카페 / 디저트,서울시 강남구 역삼동 646-9,070-7798-0544,아메리카노,3,900원,만원 미만
17.북쪽,퓨전 양식,서울시 강남구 신사동 656-6,02-542-3030,오물,포우,34,000원,4만원 이상
18.미에,이끼카페 / 호텔 / 호텔,서울시 강남구 신사동 656-7,070-0273-4087,사치미,아메리카노,10,000원,만원 이상
19.산스테이크,아메리카노,호텔,서울시 강남구 신사동 644-18,02-548-9621,한우,인제,38,000원,3만원~4
20.스치오,회 / 스키,서울시 강남구 역삼동 1-1,02-541-6200,한지 스키,오마카세,카운터,110,000원,4만원 이상
21.파에요디,프랑스 음식,서울시 강남구 신사동 646-23,02-546-7719,한지 코스,41,800원,4만원 이상
22.골드미스트,프랑스 음식,서울시 강남구 신사동 646-23,02-546-7719,한지 코스,41,800원,4만원 이상
23.비베리,기타 양식,서울시 강남구 신사동 32-3,02-6397-0997,새우,오물,스치오,15,000원,만원~2만원
24.로이,베이커리,베이커리,서울시 강남구 신사동 650-7,02-538-4000,양과,인제,18,500원,만원 미만
25.파에요디,프랑스 음식,서울시 강남구 신사동 646-23,02-546-7719,한지 코스,41,800원,4만원 이상
26.스치오,회 / 스키,서울시 강남구 신사동 651-16,02-514-0812,오마카세,카운터,110,000원,3만원~4만원
27.한도,프랑스 음식 / 호텔,서울시 강남구 신사동 650-7,02-538-4000,양과,인제,18,500원,만원~2만원
28.진미,양식,양식,서울시 강남구 신사동 115-10,02-515-3469,양식,11,000원,만원~2만원
```

{ } myjson

A simple JSON store for your web or mobile app

1 Enter your valid JSON data here.

```
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

Save

json을 복사해서 myjson.com에서 url 생성
조건에 맞게 cube 선택지에 카테고리별로 parsing

03. 안드로이드 앱 : 맛집 추천을 위한 감성분석



맛집 리뷰 데이터 감성분석(Sentimental analysis)

리뷰 텍스트를 mecab 형태소 분석기로 파싱

```
from konpy.tag import Mecab
import pandas as pd

review_data = open("../review_리뷰랜머치.txt", "r", encoding="utf-8")
review_data = review_data.read().replace("[", "").replace("]", "")

m = Mecab()
review_pos = m.pos(review_data)
```

Hannanum	Kkma	Komorana	Mecab	Twitter
아버지가방에 들어가 / N	아버지 / NNG	아버지가방에 들어가신다 / NNP	아버지 / NNG	아버지 / Noun
이 / J	가방 / NNG		가 / JKS	가방 / Noun
시~다 / E	예 / JKM		방 / NNG	예 / Josa
	들어가 / VV		예 / JKB	들어가신 / Verb
	시 / EPH		들어가 / VV	다 / Eomi
	~다 / EPN		신다 / EP+EC	

KOSAC에서 제공하는 polarity.csv 데이터를 감성사전으로 활용

	1	ngram	freq	COMP	NEG	NEUT	None	POS	maxvalue	maxprop
171	가격/NNG		3	0	0	0	0	1 POS		1
172	가격/NNG;보다/JKB		1	0	0	0	0	1 POS		1
173	가격/NNG;의/JKG		1	0	0	0	0	1 POS		1
174	가격/NNG;의/JKG;절반/NNG		1	0	0	0	0	1 POS		1
175	가까이/MAG		2	0	0.5	0	0	0.5 NEG		0.5
176	가까이/MAG;다가/VV		1	0	0	0	0	1 POS		1
177	가까이/MAG;초과/NNG		1	0	1	0	0	0 NEG		1

파싱한 리뷰 데이터를 감성사전과 비교 -> 긍정/부정 점수 부여

```
for i in range(len(ngram_comp)):
    for j in range(len(review_morphs_pos)):
        if review_morphs_pos[j] in ngram_comp[i].keys():
            review_comp_match.append(ngram_comp[i])
```

```
comp_average = (comp_sum/len(review_comp_match)) * 100
neg_average = (neg_sum/len(review_neg_match)) * 100
neut_average = (neut_sum/len(review_neut_match)) * 100
none_average = (none_sum/len(review_none_match)) * 100
pos_average = (pos_sum/len(review_pos_match)) * 100
```

```
/home/pyo/PycharmProjects/project/venv/bi
comp_sum: 242.729650257001
comp_average: 1.900185143705973
neg_sum: 4671.055637265268
neg_average: 36.56689867907678
neut_sum: 833.7322134789931
neut_average: 6.526790460928394
none_sum: 1405.7927227000057
none_average: 11.005109775324923
pos_sum: 5620.689776630054
pos_average: 44.001015943557654
```

Process finished with exit code 0

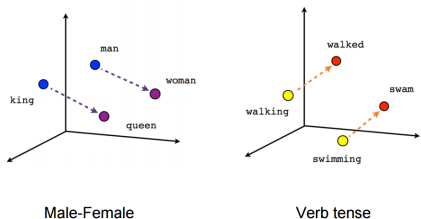
맛집 <리뷰랜머치> 결과값

긍정: 44% 부정: 36.5% 중립: 6.5% none: 11% comp: 1%

03. 안드로이드 앱 : 맛집 추천을 위한 감성분석



머신러닝(Word2vec / Naïve Bayesian Classification)



```
In [13]: from gensim.models import word2vec
model = word2vec.Word2Vec.load("review.model")
model.most_similar(positive=['디저트'])
```

C:\Users\Tj\Anaconda3\lib\site-packages\ipykernel.
ed in 4.0.0, use self.wv.most_similar() instead).
This is separate from the ipykernel package so

```
Out[13]: [('에쁘다', 0.883761644363403),
('다양하다', 0.8328750133514404),
('테이크아웃', 0.8220196962356567),
('인테리어', 0.8108780980110168),
('카페', 0.8062304258346558),
('넓다', 0.7952791452407837),
('음료', 0.7928849458694453),
('작고', 0.7916038036346436),
('무스', 0.7888064045906067),
('즐거다', 0.786755324363708)]
```

Bayesian Model

- 확률을 기반으로 어떠한 문제에 대해 결론을 내리는 모델
- Bayesian Model은 Bayesian Theory(베이즈정리)를기반으로한다.

```
In [2]: #from bayes import BayesianFilter
bf = BayesianFilter()

#테스트 학습
bf.fit("잘먹히고 맛도 좋지만 가격은 좀 있습니다.", "price_negative_favorable")
bf.fit("분위기나 서비스는 좋으나 앞에 비해 가격은 약간 부담스러운 편입니다.", "price_ne")
bf.fit("일단 맛이 없지는 않지만 가격이 너무 비싸요.", "price_negative_favorable")
bf.fit("물론 가격대가 높은 편이지만 정갈한 일대에 생겨나는 갖표요리집들과 비교해보면 착")
bf.fit("게다가 스시조의 내공의 충분히 담긴 요리들을 그의 반절도 되는 가격에 맛볼 수 있다")
bf.fit("가볍게 식사를 하기에는 가격이 부담 될 수도 있는 수준.조용하게 식사하기에 좋았다.")
bf.fit("이 가격주고 비를 해치고 먹으러 와서 했다 차과같이 좋았어요.점심 맛있게 거라고 생")
bf.fit("플러리 높은 알식집 최고는 그날 평균 가격대인듯기 원래 알식이 알식보다 더 비싼 는")
bf.fit("어느정도 가격대가 있고 고급스러운 분위기. 맛이라 그확차해 만족하지만 알이 약간")
bf.fit("가격대임을 알적을 분위기중을", "price_negative_favorable")
bf.fit("종류가 꽤나 많은데 맛그대로 양은 적고 가격대는 좀 높은편.", "price_negative_fav")
bf.fit("근데 3만원 돈주고 먹기에 가성비는 별로입니다.", "price_negative_unfavorable")
bf.fit("가성비로는 차라리 제일재민스 차를맛아우동이 낫더군요.절어소바: 달백 고소합니다.")
bf.fit("물은 아주 괜찮았고 안키모를 무수로 집과 내는 아이디어도 나쁘지 않았음. 절어소바")
bf.fit("비싸도 맛있어요", "price_negative_favorable")
bf.fit("맛이야 당연히 좋을 가성비 따지면서 먹을 곳은 아닐", "price_negative_favorable")
bf.fit("좋은맛인데 거의 18만원 가까이 나뉘는정도로 가성비는 최악.", "price_negative_un")
bf.fit("사악한 가격 가성비 떨어지는 코스메뉴", "price_negative_unfavorable")
bf.fit("비싼데 맛있어요", "price_negative_favorable")
```

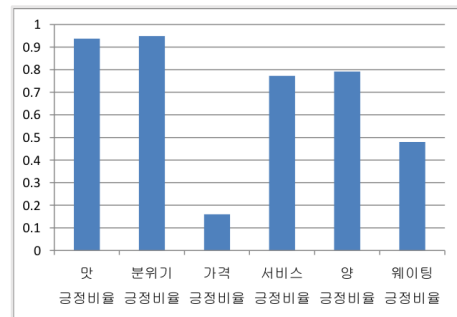
```
In [5]: #예측
pre, score_list = bf.predict("가격은 좀 있지만 괜찮았어요")
print("결과=", pre)
print(score_list)

결과= price_negative_favorable
[('price_negative_favorable', -17.008410189928963), ('price_positive_favorable', -20.53133788614623), ('price_negative_unfavorable', -21.489513817489566)]
```

How?

- word2vec을 활용해서 6개(맛, 분위기, 가격, 서비스, 양, 웨이팅)의 감성 사전을 구축한다
- 나이브베이지 분류로 감성사전을 활용해 맛집을 분류한다
- 맛집들의 세분화된 긍/부정 분류

	A	B
1	맛 감성어	감성
2	바식	POS
3	추천	POS
4	백백	NEG
5	조백	POS
6	고급스럽	POS
7	별로	NEG
8	싫어하다	NEG
9	핵초맛	POS
10	괜찮다	POS
11	눅눅	NEG
12	아쉬움	NEG
13	고소하다	POS
14	촉촉하다	POS
15	육즙	POS
16	부드럽다	POS
17	맛있다	POS



사용자에게 가장 적합한 맛집을 추천!

03. 안드로이드 앱 : 모가앱 시연 영상

MOGA



*We will keep improving our project
Thank you for watching!*

Made by

DB&챗봇

Application

정택환

정채빈

표형균

박경빈

김지현